(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－297736

(43)公開日　平成9年(1997)11月18日

(51)Int.Cl.⁶		識別記号	庁内整理番号	FI			技術表示箇所
G06F	15/00	390		G06F	15/00	390	
	3/14	330			3/14	330	A
	13/00	355			13/00	355	

審査請求　有　請求項の数32　OL　外国語出願　（全69頁）

(21)出願番号　特願平8-343551

(22)出願日　平成8年(1996)12月24日

(31)優先権主張番号　577728
(32)優先日　1995年12月22日
(33)優先権主張国　米国（US）

(71)出願人　597004720
サン・マイクロシステムズ・インコーポレーテッド
Sun Microsystems, Inc.
アメリカ合衆国カリフォルニア州94043－1100，マウンテン・ヴュー，ガーシア・アヴェニュー　2550，エムエス・ピーエイエル1－521

(72)発明者　ジョン・タン
アメリカ合衆国カリフォルニア州94306，パロ・アルト，フラワーズ・レーン　3126

(74)代理人　弁理士　湯浅　恭三　（外6名）

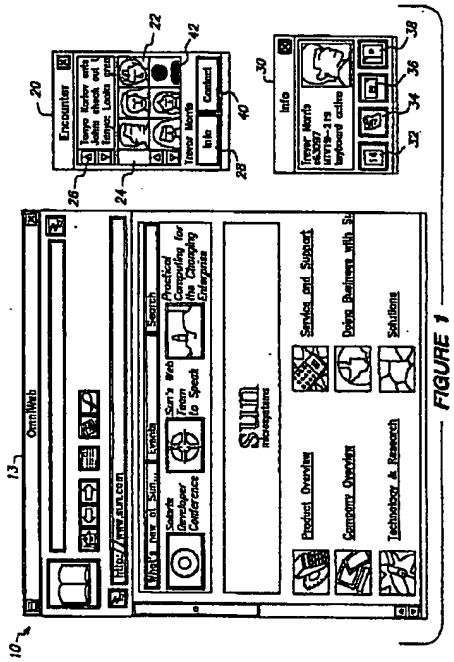
最終頁に続く

(54)【発明の名称】コンピュータ業務環境において類似のタスクを行っている他人の認識を可能にするシステム及び方法

(57)【要約】

【課題】　多数のコンピュータから構成される環境において、自分とタスク近接である他のユーザとの交流を容易にすること。

【解決手段】　他のユーザに関する情報が、それぞれのコンピュータのユーザ・インターフェース・ウィンドウに表示され、そのコンピュータ上で作業しているユーザとタスク近接であるそれぞれのユーザの視覚的な表現が示される。この情報を与える例示的なシステム・アーキテクチャは、それぞれの表し、そのユーザの視覚的な表現を記憶するパーソン・オブジェクトを含む。それぞれのコンピュータ・ディスプレイ上のエンカウンタ・ウィンドウ（２０）が視覚的な表現を与える。



1

【特許請求の範囲】

【請求項１】　複数のユーザのためのコンピュータ・システムであって、第１のユーザに、前記第１のユーザにタスク近接（task proximate）な選択された第２のユーザの視覚的な表現を提供するコンピュータ・システムにおいて、

それぞれが複数の実行可能なアプリケーションを有する、複数のコンピュータを備えており、

前記複数のコンピュータの中の１つが、第１のユーザの第１のコンピュータであり、前記第１のコンピュータは、第１のデータにアクセスする第１のアプリケーションを有し、

前記複数のコンピュータの残りのコンピュータは、それぞれが、第２のユーザの第２のコンピュータであり、前記第２のコンピュータは、それぞれが、第２のデータにアクセスする第２のアプリケーションを有し、

前記第１のコンピュータは、任意の第１のアプリケーションに対して、選択された第２のユーザの視覚的な表現を表示する第１のユーザ・インターフェース・ディスプレイを有し、前記第２のユーザは、それぞれが、前記第１のデータと第２のデータとの間の第１の関係に従って選択される、ことを特徴とするコンピュータ・システム。

【請求項２】　請求項１記載のコンピュータ・システムにおいて、前記第１のデータと前記第２のデータとの間の前記第１の関係は、前記第１のデータが前記第２のデータと同一である場合を含むことを特徴とするコンピュータ・システム。

【請求項３】　請求項２記載のコンピュータ・システムにおいて、前記第１のデータと前記第２のデータとの間の前記第１の関係は、更に、前記第１のアプリケーションによる前記第１のデータへのアクセスは前記第２のアプリケーションによる前記第２のデータへのアクセスから所定の長さの時間内である場合を含むことを特徴とするコンピュータ・システム。

【請求項４】　請求項２記載のコンピュータ・システムにおいて、前記第１のデータと前記第２のデータとの間の前記第１の関係は、更に、前記第１のアプリケーションによる前記第１のデータへのアクセスは前記第２のアプリケーションによる前記第２のデータへのアクセスと実質的に同時である場合を含むことを特徴とするコンピュータ・システム。

【請求項５】　請求項１記載のコンピュータ・システムにおいて、それぞれのアプリケーションはタイプを有しており、それぞれの第２のユーザは、更に、前記第１のユーザによって用いられている前記第１のアプリケーションのタイプと前記第２のユーザによって用いられている前記第２のアプリケーションのタイプとの間の第２の関係によって選択されることを特徴とするコンピュータ・システム。

2

【請求項６】　請求項５記載のコンピュータ・システムにおいて、前記第２の関係は、前記第１のアプリケーションが前記第２のアプリケーションと同じアプリケーション・タイプを有する場合を含むことを特徴とするコンピュータ・システム。

【請求項７】　請求項１記載のコンピュータ・システムにおいて、それぞれの第２のユーザは、更に、前記第１のユーザによって用いられている前記第１のアプリケーションが前記第２のユーザによって用いられている前記第２のアプリケーションと同一であることに従って選択されることを特徴とするコンピュータ・システム。

【請求項８】　請求項１記載のコンピュータ・システムにおいて、前記第１のデータは第１のファイル・ディレクトリに記憶された第１のデータ・ファイルであり、前記第２のデータは第２のファイル・ディレクトリに記憶された第２のデータ・ファイルであり、前記第１の関係は、更に、前記第１のファイル・ディレクトリと前記第２のファイル・ディレクトリとが同一である場合を含むことを特徴とするコンピュータ・システム。

【請求項９】　請求項１記載のコンピュータ・システムにおいて、

それぞれの選択された第２のユーザに対して、前記第１のユーザの視覚的な表現を表示する第２のユーザ・インターフェース・ディスプレイを有する前記第２のユーザの第２のコンピュータを含むことを特徴とするコンピュータ・システム。

【請求項１０】　請求項１記載のコンピュータ・システムにおいて、

それぞれのコンピュータは、２人のユーザのそれぞれに関連するデータを比較するように構成された少なくとも１つの機能的要素を含み、それに応答して、前記データが前記第１の関係を有するかどうかを示す信号を提供することを特徴とするコンピュータ・システム。

【請求項１１】　請求項１０記載のコンピュータ・システムにおいて、前記機能的要素は、更に、第１のアプリケーションのアプリケーション・タイプと第２のアプリケーションのアプリケーション・タイプとを比較し、それに応答して、前記第１及び第２のアプリケーションが同じタイプであるかどうかを示す信号を提供するように構成されていることを特徴とするコンピュータ・システム。

【請求項１２】　請求項１記載のコンピュータ・システムにおいて、第２のユーザの視覚的表現は、前記第１のユーザが前記第１の関係に従って前記第２のユーザにタスク近接になるのと実質的に同時に前記第１のユーザ・インターフェース・ディスプレイに付加され、第２のユーザの視覚的表現は、前記第１のユーザが前記第１の関係に従って前記第２のユーザにもはやタスク近接でないと決定されるのと実質的に同時に前記ユーザ・インターフェース・ディスプレイから除去されることを特徴とす

るコンピュータ・システム。

【請求項13】　複数のユーザのためのコンピュータ・システムであって、第1のユーザに、選択された第2のユーザの視覚的な表現を提供するコンピュータ・システムにおいて、

それぞれが複数の実行可能なアプリケーションを有する、複数のコンピュータを備えており、それぞれのアプリケーションはタイプを有し、

前記複数のコンピュータの中の1つが、第1のユーザの第1のコンピュータであり、前記第1のコンピュータは、第1の時間に第1のデータにアクセスする第1のアプリケーションを有し、

前記複数のコンピュータの残りのコンピュータの中の1つは、それぞれが、第2のユーザの第2のコンピュータであり、前記第2のコンピュータは、それぞれが、第2の時間に第2のデータにアクセスする第2のアプリケーションを有し、

前記第1のコンピュータは、任意の第1のアプリケーションと任意の第1のデータとに対して、前記第1のユーザにタスク近接な選択された第2のユーザの視覚的な表現を表示する第1のユーザ・インターフェース・ディスプレイを有し、前記選択された第2のユーザは、それぞれが、

前記第1のデータと前記第2のデータとの間の第1の関係と、

前記第1のアプリケーションのタイプと前記第2のアプリケーションのタイプとの間の第2の関係と、

前記第1の時間と前記第2の時間との間の第3の関係と、

を含むグループからの少なくとも1つの関係に従って、前記第1のユーザにタスク近接であるかどうかが個別に決定されることを特徴とするコンピュータ・システム。

【請求項14】　請求項13記載のコンピュータ・システムにおいて、前記第1のデータが前記第2のデータと同一である第1の関係の場合に第1のユーザは第2のユーザにタスク近接なことを特徴とするコンピュータ・システム。

【請求項15】　請求項13記載のコンピュータ・システムにおいて、前記第1のアプリケーションのタイプが前記第2のアプリケーションと同一である第2の関係の場合に第1のユーザは第2のユーザにタスク近接なことを特徴とするコンピュータ・システム。

【請求項16】　請求項13記載のコンピュータ・システムにおいて、前記第1のユーザが前記第1のアプリケーションを用いる第1の時間が、前記第2のユーザが前記第2のアプリケーションを用いる第2の時間に対して所定の長さの時間内にある第3の関係の場合に第1のユーザは第2のユーザにタスク近接なことを特徴とするコンピュータ・システム。

【請求項17】　請求項13記載のコンピュータ・システムにおいて、前記ユーザ・インターフェース・ディスプレイは、前記第1のユーザにタスク近接なすべての第2のユーザに第1の視覚的表現を提供し、他のどのユーザも前記第1のユーザにタスク近接でないときには第2の視覚的表現を提供する動作の第1のモードを含むことを特徴とするコンピュータ・システム。

【請求項18】　請求項17記載のコンピュータ・システムにおいて、

前記第1のコンピュータの前記ユーザ・インターフェース・ディスプレイは、更に、少なくとも1人の第2のユーザが前記第1のユーザにタスク近接であるかどうかに関する視覚的表現だけを提供する動作の第2のモードを含み、

それぞれの第2のコンピュータの前記ユーザ・インターフェース・ディスプレイは、前記第1のコンピュータの前記ユーザ・インターフェース・ディスプレイが動作の前記第2のモードにあることを示す前記第1のユーザに関する視覚的表現を表示することを特徴とするコンピュータ・システム。

【請求項19】　請求項13記載のコンピュータ・システムにおいて、

第2のユーザの第2のコンピュータを更に含んでおり、前記第2のコンピュータは、前記第1のユーザが前記第2のユーザにタスク近接であるという前記第1のユーザに関する視覚的表現を含むユーザ・インターフェース・ディスプレイを有し、前記第1のユーザに関する前記視覚的表現は、前記第1のコンピュータのユーザ・インターフェース・ディスプレイの表示モードの関数であることを特徴とするコンピュータ・システム。

【請求項20】　請求項13記載のコンピュータ・システムにおいて、前記第1のユーザが前記ユーザ・インターフェース・ディスプレイにおける視覚的表現によって表される第2のユーザへの通信を開始することを可能にする通信メカニズムを更に含むことを特徴とするコンピュータ・システム。

【請求項21】　請求項13記載のコンピュータ・システムにおいて、第2のユーザが前記第1のユーザに対してタスク近接になる度に第1の聴覚的な指示が提供され、第2のユーザが前記第1のユーザに対してもはやタスク近接でなくなる度に第2の聴覚的な指示が提供されることを特徴とするコンピュータ・システム。

【請求項22】　請求項13記載のコンピュータ・システムにおいて、それぞれのコンピュータは、更に、

選択された時間にユーザがアクセスしているデータを含む位置を特定するメッセージであって、前記ユーザが位置を変更するおおよその時間に、アクティブなアプリケーションによって提供されるメッセージを提供することができる少なくとも1つのアプリケーションと、

第1のユーザの第1の位置と第2のユーザの第2の位置とを比較して前記第1及び第2のユーザは相互にタスク

近接であることを示す信号を提供する少なくとも１つの一致オブジェクトと、

を備えており、前記第１の位置は、前記第１のデータ、前記第１のアプリケーション、及び前記第１の時間を含み、前記第２の位置は、前記第２のデータ、前記第２のアプリケーション、及び前記第２の時間を含むことを特徴とするコンピュータ・システム。

【請求項２３】　請求項２２記載のコンピュータ・システムにおいて、

それぞれが、このコンピュータ・システムのユーザと一意的に関連し、前記ユーザ・インターフェース・ディスプレイにおいて表示されることのできる視覚的表現を有する、複数の個人オブジェクトを更に含むことを特徴とするコンピュータ・システム。

【請求項２４】　請求項１ないし請求項２３の中の任意のコンピュータ・システムにおいて、それぞれの選択された第２のユーザの前記視覚的表現は、周期的に更新され、前記第２のユーザの利用可能性の現在のレベルを示すことを特徴とするコンピュータ・システム。

【請求項２５】　請求項１ないし請求項２４の中の任意のコンピュータ・システムにおいて、前記選択されたユーザの視覚的表現の更新は、利用可能性の第１のレベルと関連する第１の視覚的表現を、利用可能性の第２のレベルと関連する第２の視覚的表現によって代替することを特徴とするコンピュータ・システム。

【請求項２６】　請求項１ないし請求項２５の中の任意のコンピュータ・システムにおいて、前記第１の視覚的表現は、前記第２の視覚的表現に、徐々に視覚的に変換されることを特徴とするコンピュータ・システム。

【請求項２７】　それぞれのユーザがインターフェース・ディスプレイを有するディスプレイ装置を有する、複数のユーザのためのコンピュータ・システムで、第１のユーザにタスク近接である第２のユーザの認識を提供する、コンピュータによって実現される方法において、

前記第１のユーザのために、前記第１のユーザが用いている第１のアプリケーションにおける第１の位置を決定するステップと、

前記第２のユーザのために、前記第２のユーザが用いている第２のアプリケーションにおける第２の位置を決定するステップと、

前記第１の位置が前記第２の位置にタスク近接であるかどうかを判断するステップと、

前記第１の位置が前記第２の位置にタスク近接であることに応答して、それぞれのユーザのために、それぞれのユーザに関連するインターフェース・ディスプレイにおいて、他方のユーザの視覚的表現を表示するステップと、

それぞれのユーザのために、前記ユーザの視覚的表現を周期的に更新して、前記ユーザの利用可能性の現在のレベルを示すステップと、

を含むことを特徴とする方法。

【請求項２８】　請求項２７記載の方法において、選択されたユーザの視覚的表現を周期的に更新するステップは、

利用可能性の第１のレベルと関連する第１の視覚的表現を、利用可能性の第２のレベルと関連する第２の視覚的表現によって代替するステップを含むことを特徴とする方法。

【請求項２９】　請求項２７記載の方法において、前記第１の位置が前記第２の位置にタスク近接であるかどうかを判断するステップは、

前記第１の位置において特定された第１のデータと前記第２の位置において特定された第２のデータとの比較と、

前記第１の位置において特定された第１のアプリケーションと前記第２の位置において特定された第２のアプリケーションとの比較と、

前記第１の位置において特定された第１のアプリケーションのタイプと前記第２の位置において特定された第２のアプリケーションのタイプとの比較と、

前記第１の位置において特定された第１の時間と前記第２の位置において特定された第２の時間との比較と、

を含む比較群の中の少なくとも１つを更に含むことを特徴とする方法。

【請求項３０】　請求項２７記載の方法において、前記第１のデータと第２のデータとは、少なくとも１つのディレクトリ構造の中のデータ・ファイルであり、前記第１の位置において特定された第１のデータと前記第２の位置において特定された第２のデータとを比較するステップは、前記第１のデータ・ファイルが前記第２のデータ・ファイルと同じディレクトリ・レベルにあるかどうかを判断するステップを更に含むことを特徴とする方法。

【請求項３１】　請求項２７記載の方法において、前記第１のユーザの位置が前記第２のユーザの位置にもはやタスク近接ではなくなるような前記第１のユーザの位置の変化に応答して、前記第１のユーザの視覚的表現を前記第２のユーザのインターフェース・ディスプレイから除去するステップと、前記第２のユーザの視覚的表現を前記第１のユーザに関連するインターフェース・ディスプレイから除去するステップと、を更に含むことを特徴とする方法。

【請求項３２】　複数のユーザのためのコンピュータ・システムにおける、第１のユーザの第１のコンピュータによってアクセス可能なコンピュータ可読なメモリにおいて、前記第１のコンピュータは、プロセッサと表示装置とを有し、このメモリは、第１のユーザにタスク近接である第２のユーザの認識を第１のユーザに提供する前記プロセッサによって実行可能な少なくとも１つのコンピュータ・プログラムを記憶しており、前記コンピュー

タ・プログラムはプロセッサを制御して、
前記第１のユーザのために、前記プロセッサ上で実行されている第１のアプリケーションにおける第１の位置を決定し、
第２のユーザに前記第２のユーザが用いている第２のアプリケーションにおける第２の位置を示す信号を受け取り、
前記第１の位置が前記第２の位置にタスク近接であるかどうかを判断し、
前記第１の位置が前記第２の位置にタスク近接であることに応答して、前記第２のユーザの視覚的表現を、前記第１のユーザのコンピュータに関連するインターフェース・ディスプレイにおいて表示することを特徴とするコンピュータ可読なメモリ。

【発明の詳細な説明】

【０００１】

【関連出願】この出願は、「分散したワークグループ構成員へのアクセスを可能にするコンピュータ・ユーザ・インターフェースを提供するシステム及び方法」（System and Method Providing a Computer User Interface Enabling Access to Distributed Workgroup Members）と題する本願の出願人に譲渡されている先の米国特許出願に関連している。この米国特許出願は、本願で援用される。

【０００２】

【発明の属する技術分野】本発明は、一般に、コンピュータのユーザ・インターフェース設計に関係し、更に詳しくは、コンピュータに基づく作業環境においてユーザの協力を向上させるためのユーザ・インターフェース及び方法に関する。

【０００３】

【従来の技術】多くの職場では、従業員（workers）の生産性の程度は、他の従業員と直接に交流（相互作用）して、共通の問題、課題、又は関心についての情報を交換できる可能性に、大きく依存する。共通のプロジェクトにおける種々の側面を共有するチーム又はワークグループの構成員にとっては、インフォーマルな（形式張らない）交流が、極度に重要である。最近の研究によれば、インフォーマルな交流は、組織における情報の流れの中の著しい量を左右するにもかかわらず、その有効性に関して利用があまり行われていないとされる。特に、組織内における情報の流れに関する最近の調査は、組織内の人間が同じ組織の中の他の人間と情報を交換したいと考える際には、そのような機構はあまり有効ではないと考えているにもかかわらず、文書やプレゼンテーション（発表）などのフォーマルな機構を用いて、階層的な管理の連鎖を通じてコミュニケーションを行おうとする傾向にあることを示している。他方で、従業員は、その組織の中の他の部分から情報を得たいと考えるときには、個人的に知っており尊敬している従業員に質問する傾向があり、従って、インフォーマルな交流、特に、個人的な口頭でのコミュニケーションに依存している。この調査は、インフォーマルな交流が、多くの場合に、職場における共同体感覚を促進し、チームとしての一体感を高め、より効率的な問題解決に導き、従業員の間での知識や経験の共同の管理を向上させることを暗示している。知識や経験をこのように集積することにより、個々の従業員や全体としてのワークグループの成績が向上し、多くの場合に、モラール（morale）や仕事の満足度も向上する。

【０００４】物理的な近接性（proximity）は、グループの構成員が、インフォーマルで、予定していない交流を開始することを可能にすることによって、グループとしての業務をサポートする。特に、グループの構成員が近くのオフィスや業務空間を共有するような業務環境では、グループの他の構成員が働いていることを意識することが多い。これによって、グループのある構成員は、そこにいて働いているグループの他の構成員に容易に接触し、関心のある何らかの課題について会話を始めることが可能になる。他人を意識していることが、自発的で即興的な交流を実現させるのに重要であることは知られており、そのような交流によって、従業員たちは作業を調整し、共通の課題についての理解を共有することが容易になる。このように、インフォーマルに交流することの可能性は、業務に対して大きな効果を有し、従業員が他の従業員を意識していることによって、この交流が容易になり得る。

【０００５】グループの構成員は、共通の問題の種々の側面に関して仕事をする際に、実行するタスクが重複していたり、用いる情報やデータが同一のソースに由来していることが多い。グループ構成員の間でのインフォーマルな交流の多くは、これらのタスクにおける共通の問題を解決することに向けられている。これらの活動の間に同僚に遭遇することは、しばしば、インフォーマルではあるが有効な個人間のコミュニケーションの機会を提供する。更に、他の従業員を意識していることにより、共通のタスクや機能に関する課題を共有することが容易になる。

【０００６】しかし、いくつかの傾向が組み合わされて、ワークグループが密接であり続けることを困難にしている。組織のサイズが大きくなるにつれて、ワークグループの構成員は、多くの場合に、キャンパス内の異なった建物に分散されるし、更には、地球規模で、異なった地理的なサイトに分散されることになる。また、企業は、フレキシブルな業務スケジュールや、通信による、すなわち、自宅業務計画などを採用している。更に、企業の職場におけるコンピュータの使用が増大していることも、この傾向に影響している。

【０００７】これらの傾向が組み合わされ、物理的な意味で相互に密接な位置に存在していることによってワー

クグループが伝統的に共有していた、物理的なアクセスが失われている。ワークグループの数は、ネットワークに接続されたコンピュータ・システム、ファクシミリ、テレビ会議（video teleconferencing）などの電子的に仲介される機構によって、増加している。これらの電子的な設備は、ワークグループを分断しかねない物理的な距離を超えて生の情報が流通することを非常に効果的に助けるが、物理的に同じ場所で働く人間であれば共有している相互の意識感覚や交流の機会を、同じ豊かさにおいて与えることはできない。

【０００８】従って、分散しているワークグループの構成員に対して、物理的に相互に近接した位置にいる従業員であれば共有する意識の感覚に類似の感覚を与えるコンピュータに基づく機構（メカニズム）を提供することが望まれる。望ましい共同的なコンピュータ・システムにおいては、アプリケーションは、タスクが近接している（task proximate、以下では「タスク近接」と称する）他の人間に関する意識を適切に提供しなければならない。従業員は、同じ又は関連するデータに関して、同じ又は関連するアプリケーションを用いて、ほぼ同じ時刻に業務を行っている場合に、「タスク近接」であるとする。所望の機構は、更に、タスク近接である他の者や交流したいと考える者が、会話を始める又は何らかの形で出会う方法を提供する。

【０００９】多くの種類のコンピュータによって実現される通信装置が知られているが、これらの既知の装置では、任意の種類のタスクに関して類似のタスクを有する他の従業員についての意識を生じさせる一般化した機構を提供することはできていない。従来の製品の中にも、特定のタイプのアプリケーションに向けた非常に限定された情報を与えるものは存在している。例えば、ワールド・ワイド・ウェブ（WWW）にアクセスするためのウェブ・ブラウザは、２人又はそれより多くのユーザが同時に同じウェブ・ページ上にいるかどうかだけを判断できる。その際には、両方のユーザは同じウェブ・ブラウザ・アプリケーションを有していなければならず、そのウェブ・ページは、それらのウェブ・ブラウザと互換性のあるウェブ・サーバによって提供されなければならない。これによって、他人に関する意識は、ユーザと計算資源に関する非常に小さな特定のコミュニティに限定されてしまう。更に、この構成では同じウェブ・ページに複数のユーザがいるかどうかだけを判断するように設計されているので、そのような製品では、種々の異なるアプリケーションと共に用いて、複数の個人が種々の使用時期において関連するデータに関して関連するタスクを実行しているかどうかを判断することのできる一般化されたシステム・アーキテクチャを提供することはできない。更に、そのような製品では、異なる従業員が実行しているタスクの間の関係を示し彼らの間のコミュニケーションを容易にする一般化されたユーザ・インターフェースを提供できない。

【００１０】従来の製品には、これ以外にも、地理的なモデルに依拠して、ユーザのコミュニティが共有している環境をシミュレートするものもある。その例として、マルチ・ユーザ・ダンジョン（multi-user dungeon）すなわち、ＭＵＤや、オブジェクト指向（object-oriented）のＭＵＤ、すなわち、ＭＯＯなどがある。これらの機構は、仮想的な空間を提供し、ユーザは、その仮想的空間を通って進み、その中で同じ部屋又は位置を共有する他のユーザと交流する。これらの機構は、仮想空間の、タスク指向（task-oriented）のモデルではなく、位置指向（location-oriented）のモデルに完全に基づいている。更に、ＭＯＯ及びＭＵＤは、他のアプリケーションがそれと共に動作しタスク近接情報を提供するアーキテクチャではなく、それ自身の環境、ある意味では、それ自身のアプリケーションとしての使用のために設計されている。更に、ほとんどのＭＯＯ及びＭＵＤは、エンターテイメントの目的、特に、他のユーザに会う目的で用いられる。

【００１１】他にも、方向の決まった通信能力を有するだけの従来型のコンピュータ・ツールが存在する。例えば、電子メール又はテレビ会議用の製品によれば、ユーザは、特定のモードにおいて、他の従業員と直接に通信できる。しかし、これらの製品は、ユーザのタスク近接性については、何の情報も提供しない。むしろ、これらのツールは、他のユーザが１人又は複数の特定の人間と通信することを望んでいることを既に知っているユーザのためのものである。従って、これらの製品は、相互にタスク近接であるユーザを意識することによって可能になる自発的なタイプの交流を実現することはできない。

【００１２】

【発明の概要】本発明の１つの実施例は、これによれば、自分のコンピュータを用いている従業員が、アクセスしているデータや、用いているアプリケーションや、そのような業務が実行された時刻などの、自分たちがしている仕事のタイプに関して、他のどの従業員が近くにいるかを知ることができる機構（メカニズム）を提供する。ある実施例では、この関係は、「タスク近接性」として知られている。ある従業員が別の従業員と「タスク近接」であるとは、両者が、特定の時間周期の間に、類似するタイプのデータにアクセスしている、又は、類似するアプリケーション・ツールを用いている場合である。データ、ツール、及び時間周期の間の関係は、異なるタイプのデータ及びアプリケーションに対応するために、変動することがある。タスク近接性の関係は、従業員が共通のタスクに関する情報を共有している物理的な業務環境において生じるタイプの交流に対応する。本発明によれば、従業員は、タスク近接である他の従業員を知ることになり、物理的な業務環境において見られるタイプの自発的な交流を容易にする。その理由は、コンピ

ュータによって実現されるタスクを行っている従業員は、他の従業員が同様のタスクを行っているという意識（awareness）から利益を得るからである。ある従業員が他の従業員をこのような態様で意識するときには、彼らのタスク又は他の何らかの関連するタスクをサポートする交流が生じる可能性が増加する。

【００１３】従業員は、典型的には、多数の異なるコンピュータ・アプリケーションを用いて、自らの機能や目的を果たす。異なる従業員が異なるアプリケーションを用いながらも、彼らは、同一の又は類似するタスクを行う場合がある。従って、本発明の別の側面によれば、異なるマシン上で、別の時刻に別のアプリケーションを用いて作業を行う異なるユーザの間でのタスク近接性を判断するシステム及びアプリケーション・アーキテクチャが、提供される。更に、最終的な目的はタスク近接であるユーザの間の交流を容易にすることであるから、本発明のアーキテクチャは、タスク近接なユーザの間でのコミュニケーションを効率的に単純に開始させる機構を提供する。最後に、本発明は、タスク近接な従業員を表すユーザ・インターフェースを与え、それぞれの従業員が、他の従業員と種々のタイプの交流を容易に開始できるようにする。

【００１４】

【発明の実施の形態】

遭遇ウィンドウ及びタスク近接性

図１を参照すると、タスク近接である他のユーザの意識を可能にする機構のユーザ・インターフェースの１つの実施例が示されている。図１には、特定のユーザの、通常は「デスクトップ」と称されるスクリーン表示が図解されている。この特定のユーザは、ここでは、カレント・ワーカ（current worker）と称することにする。このデスクトップは、カレント・ワーカのコンピュータの表示画面上に生じる。デスクトップ１０では、ブラウザ・アプリケーションのためのアプリケーション・ウィンドウ１３と、遭遇メカニズム（機構）のユーザ・インターフェース部分であるエンカウンタ（encounter、遭遇）ウィンドウ２０とがある。エンカウンタ・ウィンドウ２０は、他のどの従業員がタスク近接であるかをカレント・ワーカに知らせる視覚的な機構を与える。この遭遇メカニズムは、更に、タスク近接である従業員に関する音声的な指示も与える。タスク近接である従業員に対しては、この遭遇メカニズムは、そのような従業員との交流を効率的に開始する手段を与える。

【００１５】カレント・ワーカにタスク近接なそれぞれの従業員に対しては、エンカウンタ・ウィンドウ２０は、その従業員の適切な表現を表示する。この好適実施例では、この表現は、アイコン２２である。アイコン２２は、従業員のビット・マップされたイメージである。別の実施例では、それぞれの従業員が利用可能なハードウェア・サポートのレベルや、利用可能なネットワークの帯域幅や、それぞれの従業員が望むプライバシのレベルなどに応じて、種々のグラフィック・イメージ・フォーマットや、リアルタイムの映像や、単純なテキスト列や、それ以外の情報などの、他の形式の表現を用いることができる。ビデオ・カメラを備えているコンピュータを有する従業員の場合には、彼ら自身のビデオ・イメージをキャプチャすることによって、表現を作成してもよい。ビデオ・サポートのない従業員の場合には、アイコンをアイコンの組の中から選択したり、その従業員がアイコンを作成したり、又は、テキスト列を用いてもよい。

【００１６】エンカウンタ・ウィンドウ２０は、カレント・ワーカに対して新たな従業員がタスク近接になり別の従業員がタスク近接性を失うにつれて、周期的に更新される。カレント・ワーカ又は他の従業員の内容（コンテンツ）がアプリケーションの間で切り換わるので、エンカウンタ・ウィンドウ２０は、更新されて、その時点でカレント・ワーカにタスク近接である従業員のアイコン２２を表示する。遠隔のコンピュータで作業している従業員があるアプリケーションを用いてカレント・ワーカに対してタスク近接になるときには、エンカウンタ・ウィンドウ２０は、同じ視覚的／音声的な合図（cue、キュー）を発して、エンカウンタ・ウィンドウ２０内に、その従業員に対するアイコン２２が現れる。その人間がもはやタスク近接ではなくなると、アイコン２２は、除去される。エンカウンタ・ウィンドウ２０は、こうして、他の従業員のタスク近接性に変化があったことを、カレント・ワーカに僅かに知らせる。

【００１７】タスク近接性は、３つの判然としたファクタの中の任意のものに基づく。すなわち、１）その従業員が現に用いているアプリケーション、２）その従業員がアクセスしている又は操作しているデータ、及び３）そのような作用が生じた時刻、の３つである。これらのファクタは、様々に用いられて、「タスク空間」におけるユーザの位置を定義する。タスク近接性の最も厳密な定義は、２人の従業員が、同時に、同一のデータ・ファイルに関して、同一のアプリケーションの、同一の機能を用いている場合である。

【００１８】位置及びタスク近接性の定義は、上に掲げたファクタのそれぞれに沿って、独立に緩和することができる。例えば、

＊アプリケーションの制約を緩和して、異なるアプリケーション又はアプリケーションのタイプを用いて同じデータを見ている従業員は依然としてタスク近接である、とすることができる。例としては、異なるウェブ・ブラウザを用いて同じWWWページを見ている、又は、異なるデータベース・アプリケーションを用いて同じデータベース・テーブルにアクセスしている場合を含む。また、アプリケーションの制約を強化して、たとえ同じアプリケーションにおいても、２人の従業員は、ワード・

プロセッサにおいてスペル・チェック機能をもちいたり、コンパイラにおいてコードをコンパイルしたりなど、同じ機能を実行していなければならないようにもできる。この場合には、同じアプリケーションは用いているが別の機能を実行している従業員は、タスク近接ではないと考えられる。

【0019】＊時間に関する制約を緩和して、相互に所定の時間周期の間に同じデータにアクセスしている従業員はタスク近接であるようにすることができる。例えば、15分の間に同じ株式相場にアクセスする従業員は、相互にタスク近接であり得る。同様に、1時間の間に同じウェブ・ページ又は電子メール・メッセージにアクセスする従業員や、1日の内に同じワード・プロセシング文書にアクセスする従業員もまた、タスク近接とすることができる。

【0020】＊データ関する制約を緩和して、同じアプリケーションを用いて別のデータにアクセスしている従業員をタスク近接であるとすることができる。例えば、あるファイル・ブラウザを用いて同じファイル・ディレクトリの中の別のファイルにアクセスしている従業員をタスク近接であるとすることができる。更に別の例として、特定の日付の特定の人間のカレンダを見ている従業員を、同じ人間のカレンダを別の日付に関して見ている別の従業員とタスク近接であるとすることができる。同様に、選択された日付に関してカレンダを見て会議室などの施設を予約する従業員は、その日付に関してカレンダを見て別の会議室を予約する別の従業員とタスク近接であるとし得る。

【0021】タスク近接性は、好ましくは、現にアクティブである、又は、直近の時点でアクティブであったアプリケーションに関して判断される。タスク近接性は、アプリケーションによって、又は、ユーザの位置に関する情報を有するそれ以外のソフトウェア・コンポーネントによって、判断できる。この情報は、好ましくは、アプリケーションによって、与えられる。この情報を与えることのできるアプリケーションは、「遭遇を意識している」（encounter-aware）。与えられたタイプのアプリケーションが、同じ又は類似する態様でタスク近接性を判断することが好ましい。アプリケーションのタイプは、一般には、例えば、データベース、ワード・プロセッサ、メーラ、コード・コンパイラなどの、使用の性質又は領域に基づく。複数のタイプを有すると考えられるアプリケーションも存在する。与えられた1つのタイプの2つのアプリケーションは、それぞれのユーザに対して、実質的に同じタスク近接性の判断がなされることが好ましい。

【0022】図2は、種々のタスク近接性の関係を図解している。図2は、上述した3つのファクタによって定義される軸を有するタスク空間を表している。このタスク空間においては、個々の従業員は、ラベルの付いた円によって表されている。アプリケーション軸に沿っては、3つの遭遇意識アプリケーションL、M、Nが存在する。データ軸に沿っては、3つのデータ・ファイルX、Y、Zが存在する。時間は、連続的である。従業員A及びBは、共に、アプリケーションNを用いて、同時にデータ・ファイルZにアクセスしている。NとNのタイプの他のアプリケーションがタスク近接性に関するこの定義を強制する場合には、A及びBは、それぞれの表現を、自らのエンカウンタ・ウィンドウにおいて見ることになる。

【0023】従業員C及びHは、アプリケーションLの中におり、このLは、データ及びアプリケーションに関する制約を緩和する。アプリケーションLは、2人の従業員は、両者がアプリケーションLを用いているか、又は、同じアプリケーションを用いていない場合には同じデータにアクセスしている場合に、タスク近接であるという近接性の定義を適用する。アプリケーションLは、時間に関しては、厳密な要件を課す。従って、従業員C及びHはタスク近接であり、その理由は、彼らが、別のデータにアクセスしているが同じ時刻にアプリケーションLの中にいるからであり、彼らは、共に、互いのエンカウンタ・ウィンドウの現れる。更に、アプリケーションLは、従業員Hは従業員A及びBとタスク近接であると考える。その理由は、従業員A及びBは、異なるアプリケーションは用いているが、Hと同時に、Hと同じデータ・ファイルZにアクセスしているからである。この状況の例は、アプリケーションL及びNが別のWWWブラウザであり、データ・ファイルZがあるWWWページである場合である。別の例としては、従業員G及びFが、異なっているが関連するデータ・ファイルに関して、同時にアプリケーションMを用いている場合であり、互いのエンカウンタ・ウィンドウに現れることになる。

【0024】ある好適な実施例では、タスク近接性は、対称的であり、それによって、第1の従業員が第2の従業員のエンカウンタ・ウィンドウに現れる場合には、第2の従業員は、第1の従業員のエンカウンタ・ウィンドウに現れる。これは、ラーキング（lurking、潜行）を防止するためには、望ましい。

【0025】しかし、ある場合、例えば、時間に関する制約を緩和する際には、非対称が好ましい又は必要である。例えば、従業員D、F及びGが同一のアプリケーションを要求するタスク近接性の定義を用いるアプリケーションMを用いており、時間の制約を部分的に緩和し、データの制約を緩和するとする。この場合には、従業員D及びGは、共に、データ・ファイルにアクセスするが、異なる時間でのアクセスである（例えば、Dは、Gのアクセスから少し遅れてデータ・ファイルXにアクセスする）。すると、従業員Gは、従業員Dのエンカウンタ・ウィンドウには現れるが、Dは、Gのエンカウンタ

・ウィンドウには、現れない。

【００２６】従業員Ｅは、アプリケーションＭにおいて従業員Ｇ及びＦが行うのと同時に、アプリケーションＮを用いてデータ・ファイルＹに作用する。しかし、彼らは、それぞれのエンカウンタ・ウィンドウには現れない。その理由は、アプリケーションＭ及びＮは、共に、アプリケーションの一致を要求するからである。これらは、可能なタスク近接性の関係のタイプのいくつかの例に過ぎない。

アプリケーションのタイプ
テキスト・エディタ／ ワード・プロセッサ
ファイル・ブラウザ
電子メール・アプリケーション
コード開発環境
カレンダ・ブラウザ
ウェブ・ブラウザ

この例は、単に、異なるタイプのアプリケーションに対してタスク近接性を定義する種々の方法を図解しているに過ぎない。これらのクラスの中の任意のものに対して、時間の制約は適用されていないが、その理由は、インストールされたものに基づくサイズに応じて、従って、与えられたデータ・ファイルに対して与えられた時間周期においてタスク近接になり得る従業員の数又はそれ以外のファクタに応じて、程度を変動させることにより緩和することができるからである。

【００２９】好適実施例においては、遭遇メカニズムは、他の従業員についての「意識」の種々のモードを提供する。これらのモードによれば、カレント・ワーカは、その遭遇メカニズムが彼又は彼女の経験に入り込む程度を制御することができる。意識の程度は、それぞれの従業員によって独立に特定され、遭遇メカニズムを両方の場合において、すなわち、この遭遇メカニズムがカレント・ワーカにどのように現れるか、また、カレント・ワーカが、他の従業員のエンカウンタ・ウィンドウ２０においてどのように現れるかを、制御する。意識のモードは、デスクトップ上のすべてのアプリケーションに適用されるものとして特定される、又は、それぞれのアプリケーションにおいて個別に特定される。遭遇メカニズムの好適実施例においては、遭遇メカニズムに対して、３つのモードの意識がある。すなわち、開いたモード、極小（minimal）のモード、及び閉じたモードである。意識のモードは、メニュから、又は、モード・アイコン１８をクリックすることによって選択される。

【００３０】意識の開いたモードは、タスク近接な他の従業員との交流を受ける従業員によって用いられる。遭遇メカニズムが「開いている」として特定される際に

＊【００２７】以上で述べた例は、個々のアプリケーションに言及しているが、タスク近接性の定義は、好ましくは、そのアプリケーションのそれぞれの場合の特定の実現とは無関係に、与えられたタイプのアプリケーションの内部で一貫している。アプリケーションのクラスに関するタスク近接性の定義の例には、次のようなものを含む（しかし、これらに限定はされない）。

【００２８】

タスク近接性の定義
同じデータ・ファイル
同じディレクトリ・ツリー、又は、 １つのディレクトリ・レベルの内部
同じメッセージ
同じレベルのコード・ツリーでの コード・ファイル
同じカレンダ・ファイル、又は、 同じ日付
同じウェブ・ページ

は、従業員は、タスク近接である従業員の表現を見るし、更に、これらの表現は、従業員のタスク近接性が変化するにつれて更新される。ある従業員が遭遇メカニズムを開いたモードに設定すると、その従業員の視覚的な表現が他の従業員のエンカウンタ・ウィンドウに現れる。更に、開いたモードでは、電子メールのアドレス、電話番号、ファックス番号、住所などの、そのユーザに関する追加的な情報を得て、テレビ会議、電子メールによるメッセージなどの手段により、これらの従業員の中の任意の者との交流を開始する。

【００３１】エンカウンタ・ウィンドウにおける従業員の視覚的な表現に加えて、ある従業員がタスク近接になったり又はタスク近接性を失う度に、音声による指示を用いることもできる。これらの事象のそれぞれに対して、異なる音声的な指示を用いることができる。例えば、ある従業員がタスク近接になる際には長いビープ・トーン（beep tone）を用い、タスク近接でなくなる際には、短いビープ・トーンを用いるなどである。他の音声効果を用いてもかまわない。

【００３２】好適実施例においては、エンカウンタ・ウィンドウ２０へのアイコン２２の追加やエンカウンタ・ウィンドウ２０からのアイコン２２の除去は、従業員がタスク近接になったりタスク近接でなくなったりする際に、アイコン２２の視覚的な変換によってなされる。この視覚的な変換は、フェード（消える）、ワイプ（取り去る）、ディゾルブ（解消する）、プル（引く）、又は、アイコン２２を目立たないように配置及び除去する他の徐々に生じる視覚的な効果を含む。このような視覚的な変換は、従業員が物理的な職場で典型的に有する従業員の出入りに関する意識が、漸次に取得されること

を、より自然に反映している。

【００３３】いつの時点でも、カレント・ワーカに対してタスク近接である従業員の数は、１日の時間の結果として、アプリケーションに基づいて、ネットワークのサイズによって、あるいはそれ以外のファクタによって、大きく変動する。また、アプリケーションによっては、映像放送によるプレゼンテーション・ソフトウェアのように、多数の異なるユーザを同時に含むことを意図するものもある。これらの変動に対応するために、ユーザは、エンカウンタ・ウィンドウの外見（appearance）を制御することができる。好ましくは、遭遇メカニズムが開いているときには、エンカウンタ・ウィンドウ２０が現れる方法は、少なくとも２つ存在する。比較的少数のタスク近接である従業員が存在することが予測される、又は、実際にそうであるようなアプリケーションでは、エンカウンタ・ウィンドウ２０は、これらのユーザの表現を、コンパクトなウィンドウ・フォーマットで表示する。図１に図解されているエンカウンタ・ウィンドウ２０は、コンパクトなフォーマットを有する意識の開いたモードを用いて示されている。

【００３４】コンパクトなフォーマットでは、エンカウンタ・ウィンドウ２０は、スクロール可能なウィンドウ枠（pane）２４の中に、タスク近接な従業員の表現を含む。スクロール可能なテキスト領域２６は、どの従業員がタスク近接になり、どの従業員が「去った」のかに関する情報を与える。更に、共有されているテキスト領域２６によって、従業員が、相互にテキストによる対話によって、相互に通信することが可能になる。

【００３５】コンパクトなフォーマットは、更に、タスク近接な従業員に関する情報を取得する手段と、１人又は複数の従業員とコンタクトをとる手段とを提供する。図１では、情報ボタン２８は、タスク近接である従業員の中の１人の選択された表現に対する、ビジネス・カード（名刺）３０と称される別のウィンドウをもたらす。ビジネス・カードは、その従業員とコンタクトをとるのに有用な情報、例えば、その従業員の内線番号、メール・ストップ、ファクシミリ番号、電子メール・アドレスなどを含む。好適実施例では、ビジネス・カード３０は、地球規模でアクセス可能なデータベースに記憶された、従業員に関する情報の特定の部分を示している。

【００３６】頻繁に、従業員は複数のアプリケーションを走らせることはあるが、ある理由で、コンピュータを用いていないこともある。遭遇メカニズムの目的の１つはカレント・ワーカがタスク近接である従業員と成功裏に通信できる可能性を増加させることであるから、遭遇メカニズムは、他の従業員が現にコンピュータを用いているかどうかについての情報を提供する。この情報を提示する１つの方法は、ビジネス・カード３０においてであり、ビジネス・カードには、図１に示されているように、その従業員のキーボードなどの入力装置が現在アクティブであるかどうかを示すテキスト・データが含まれる。その従業員の入力装置がアクティブでない場合には、遭遇メカニズムは、最後にアクティブであったときからの時間の長さか、又は、最後にアクティブであった時刻を提供する。

【００３７】ビジネス・カード３０は、更に、その従業員に関する他の情報、例えば、その従業員のカレンダ３２などへのアクセスを提供する。カレント・ワーカは、その従業員の暇な時間や、タスク近接になってからの時間などの情報にもアクセスする。ビジネス・カード３０は、更に、カレント・ワーカに、特定の従業員が交流している他の従業員のアイデンティティを知らせるのに有用であり、それによって、カレント・ワーカが、その従業員との通信を試みる際の、礼儀と成功の可能性とを評価することが可能になる。

【００３８】更に、カレント・ワーカは、任意の他の従業員との通信（コミュニケーション）を開始することができる。この交流には、スティック・アップ・ボタン３４を介してそのビジネス・カードを所有する従業員にテキスト・メッセージを送ることや、電子メール・ボタン３６を介してその従業員に電子メールを送ることや、テレビ会議ボタン３８を介してその従業員とテレビ会議を始めることなどが含まれる。これらの交流フォーマットは、それぞれが、オペレーティング・システム又はアプリケーション・フレームワークにおける適切なサービス・インターフェースによって提供される。ある実施例では、これらの種々のサービスは、オブジェクト・リクエスト・ブローカと交流してあるユーザのコンピュータ上の通信アプリケーションを第２のユーザのコンピュータに結合する通信サーバ（テレビ会議サーバなど）によって管理される。

【００３９】更に詳しくは、コンタクト・ボタン４０によって、カレント・ワーカが、１又は複数の選択された従業員との交流を開始することが可能になる。ある実施例では、カレント・ワーカは、エンカウンタ・ウィンドウの中の表現から１又は複数を選択して、コンタクト・ボタン４０を押す。次に、遭遇メカニズムが、通信サーバと連携して、テレビ会議、電子メール、テキスト・チャット、音声などの通信メカニズムを、それぞれの従業員が利用可能なハードウェア／ソフトウェア・サポートに応じて、選択する。ただ１人の従業員しかテレビ会議の設備を有しておらず、他方は音声サポートだけの場合には、音声だけの会話が開始される。

【００４０】カレント・ワーカに対して多数の従業員がタスク近接であるアプリケーション又はコンテキストに対しては、エンカウンタ・ウィンドウ２０は、拡張されたフォーマットで動作される。図３には、拡張されたフォーマットの１つの例が図解されている。拡張されたフォーマットは、ウェブ・ブラウザ、データベース、又はネットワーク放送アプリケーションなどの多くの従業員

が同時に用いることがあるアプリケーションに対して有用である。

【００４１】拡張されたフォーマットでは、追加的な機能が与えられて、カレント・ワーカがタスク近接な従業員を見つけてその表現を組織化する能力を強化する。図３では、エンカウンタ・ウィンドウ２０は、多数のアイコン２２を表示するように構成され、カレント・ワーカは、無線ボタン２９を用いて、名前の表示に切り換えることができる。これにより、図４の表示が生じる。次に、カレント・ワーカは、従業員の組を、従業員の名前、従業員がタスク近接になった時刻（最も最近か、又は、その逆か）、従業員の暇な時間、又は交流活動などを含む任意の数のキー２７によって、ウィンドウ枠のトップに現れている従業員と相互に交流している従業員と、ソートする。この最後のソートのキーによって、遭遇メカニズムの協同的な性質が強化され、それぞれの従業員が、タスク近接である他の従業員と、交流している従業員との両方を見ることが可能になる。従業員は、また、名前、位置、所属などの情報を含む個人的な情報、又は、活動レベルなどのそれ以外の情報の種々のサーチ・キーを用いて、再び、特定の従業員をサーチする。

【００４２】エンカウンタ・ウィンドウのフォーマットは、特定の機能に関連している。どのアプリケーションでもユーザの数は変動するので、カレント・ワーカは、エンカウンタ・ウィンドウ２０のフォーマットを切り換えて、機能を制御する。これは、追加的な従業員がカレント・ワーカの個人的なタスク空間に「入り」（タスク近接になり）、又は、その空間から「出る」（タスク近接でなくなる）からである。フォーマットの間での切り換えは、適切なユーザ・インターフェース又はキーストロークを用いてエンカウンタ・ウィンドウ２０のサイズを決め直すことを含む多数の方法で、実行できる。

【００４３】開いたモードの間にエンカウンタ・ウィンドウ２０のフォーマットを制御することに加えて、ある実施例では、その従業員は、タスク近接性のディスプレイの感度を制御して、それにより、与えられた従業員が同時に同じデータにアクセスしているのか、異なる時刻に同じデータにアクセスしているのか、同時に異なってはいるが関連するデータにアクセスしているのかなどを、ユーザが区別できるようになる。これにより、ユーザは、それぞれの従業員のタスク近接性の意義を評価できる。タスク近接性の程度は、アイコン２２上の異なる色の境界、異なる境界のパターン、アイコン２２の位置、又は、それ以外の視覚的な属性によって、示すことができる。例えば、タスク近接性の程度の指示として位置を用いると、「より近い」（よりタスク近接な）従業員は、エンカウンタ・ウィンドウのトップに配置され、「より遠い」（よりタスク近接でない）従業員は、ボトムに配置される。

【００４４】ある場合には、従業員は、特定のタスクに集中するために、又は、それ以外の理由のために、タスク空間において、他人に関して単に周縁的にだけ意識したいことがあり得る。カレント・ワーカは、その場合には、遭遇メカニズムを、その極小モードに設定する。極小モードでは、遭遇メカニズムは、少なくとも１人の従業員がタスク近接であるかどうかだけを示し、タスク近接である従業員が誰であるかどうかは示さない。この指示は、単純な比較的小型のアイコンによって与えられる。図５ａ及び図５ｂは、極小モードの例を図解しており、極小モードのアイコン１８が、ウェブ・ブラウザ・アプリケーションのウィンドウ１３のトップに表示されている。図５ａは、人のアイコンが空になっていることで、タスク近接である従業員は他に誰もいないことを示す極小モードのアイコン１８を図解している。カレント・ワーカが別の従業員に対して最初にタスク近接になると、極小モード・アイコン１８は更新され、図５ｂに示されるように、人が存在することを暗示するようになる。他の従業員がカレント・ワーカのタスク空間に入ることは、アイコン更新の原因にならない。最後の従業員がカレント・ワーカに対するタスク近接性を失うと、アイコン１８は、図５ａに示されるように、そのブランクの状態に戻る。更に、極小モードは、オプションで、視覚的な指示に釣り合う音声的な指示を含み、従業員がタスク空間に入るときと、出るときとに対して、区別できるトーンを対応させる。

【００４５】最後に、カレント・ワーカが、他のどの従業員を意識するに全く興味をもたないこともある。従って、閉じたモードが提供され、この場合には、エンカウンタ・ウィンドウ２０は、全く用いられず、カレント・ワーカは、他の従業員のタスク近接性に関する情報を全く受け取らない。

【００４６】好適実施例では、従業員がエンカウンタ・ウィンドウにおいて何を見るかを制御することに加えて、このモードは、他の従業員が、この特定の従業員に関して何を見るかを制御する。開いたモードでは、従業員の表現が、イメージ、映像、又はそれ以外であるかによらず、タスク近接である他の従業員のエンカウンタ・ウィンドウ２０に提供される。表現は、図１に示されているようになり、種々の従業員のイメージが示される。

【００４７】極小モードでは、従業員は、他の従業員について、単に極小的に意識することを望み、この望みが、タスク近接である他の従業員に伝えられる。従って、極小モードの従業員は、図１に示されているように、シルエット４２のように、極小モードを表すアイコン又は他のイメージによって、他の従業員のエンカウンタ・ウィンドウ２０において、見られる。

【００４８】最後に、従業員が極小モードにあると、タスク近接である他の従業員には何の表現も与えられず、これは、第１の従業員の、この状態の情報を受け取らず又は送らないという意図に沿うことになる。従って、遭

遇メカニズムのモードは、好ましくは、他の従業員に与えられる、及び、他の従業員から受け取られる情報に関して、対称的である。または、従業員は、それぞれのモードに関して、他の従業員に戻される情報のタイプ又は程度を特定することができる。

【００４９】すべてのモードは、従業員が用いているすべてのアプリケーションに対して、又は、それぞれに対して個別的に、特定することができる。例えば、従業員は、ウェブ・ブラウザに対して開いたモードであることを希望し、そのウェブ・ページにどの他の従業員がタスク近接であるのかを知ることを望むことがあり得る。理由は、これによって、その従業員が関係している活動又はタスクについての更に有用な情報が得られるからである。しかし、スプレッドシートやデータベースなどのツールでは、この従業員は、極小モードでいることを望み、誰かが類似のタスクに関して作業しているかどうかだけを知りたいと考えることがある。最後に、例えば、ワード・プロセッサのような別のアプリケーションに関しては、この従業員は、関わられることを一切望まないこともある。

【００５０】本発明の別の実施例では、タスク近接である従業員の利用可能性に関する更なる情報が、アイコン２２の変化によって、エンカウンタ・ウィンドウ２０において、表示される。利用可能性は、少なくとも５つのレベルに分割され、すなわち、アクティブ、暇、関係している、邪魔するな、不在である。利用可能性のこれらのレベルのそれぞれに関しては、デフォルトのアイコン又はユーザの定義によるアイコンが割り当てられ、それぞれが、利用可能性の関連するレベルを、視覚的に判然と、暗示又は指示する。利用可能性の判然としたレベルに対応する例示的なアイコン２２ａからｅが、図９に示されている。利用可能性のレベルは、ある実施例では、例えば、キーボードの活動をモニタして、単位時間当たりのキーボード入力を利用可能性の現在のレベルを判断する計量として用いることによって、従業員のコンピュータ使用の関数として、決定される。更に、従業員のコンピュータとコンピュータ・ネットワークの通信アーキテクチャとの相互接続とを用いて、従業員が別の従業員との交流（例えば、電話、テレビ会議など）に関係しているかどうかが判断される。手続的には、ある従業員がタスク近接であると判断された後で、その従業員のアイコン２２は、利用可能性の現在のレベルを反映するように、更新される。従業員がタスク近接であり続ける限りは、そのアイコン２２は、周期的に更新されて、利用可能性の現在のレベルを反映する。

【００５１】例えば、典型的な場合には、第２の従業員が、利用可能性のアクティブなレベルにありながら、第１の従業員とタスク近接になる。従って、第２のユーザのアイコン２２は、図９にあるように、第１の従業員のエンカウンタ・ウィンドウ２０の中にアイコン２２ａとして、図解される。第２のユーザが依然として第１のユーザにタスク近接であるある時間周期の後で、第２の従業員が暇になる場合には、アイコン２２ａは、利用可能性の現在のレベルを反映するように更新されて、アイコン２２ｂのようになる。後に、第２の従業員が別の従業員との交流を開始すると（その別の従業員が第１又は第２のユーザとタスク近接であるかどうかには関係なく）、第２の従業員のアイコンは、再び更新され、例えばアイコン２２ｃになる。

【００５２】好ましくは、利用可能性アイコン２２のレベルの変化は、フェード、ワイプ、ディゾルブなどの、漸次的な視覚的な変換を用いて生じ、変化が目立たないようにしている。利用可能性情報のレベルの使用及び実現は、上述した関連出願に、更に記載されている。

【００５３】遭遇メカニズムのアーキテクチャ

次に、図６を参照すると、遭遇メカニズムを与えるシステムのブロック図が示されている。このシステムは、ネットワーク１２３上で接続された多数のコンピュータを含む。それぞれのコンピュータ１０１は、プロセッサ１０３と、アドレス指定可能なメモリ１０５と、ディスプレイ１０７と、ローカルなハードディスク１０９と、入力／出力ポート１１１と、ネットワーク・インターフェース１１３と、を含む。それぞれのコンピュータ１０１は、更に、好ましくは、その入力／出力ポートに結合された、従来型のマウス１１９などのポインティング・デバイスを有する。更に、コンピュータ１０１は、マイクロフォン１１７及びスピーカ１１８を介しての音声機能と、ビデオ・カメラ１２１を介してのビデオ機能とを有する。ユーザは、ネットワーク・インターフェース１１３を介して、ＬＡＮやＷＡＮなどのネットワークに接続して、ネーミング・サービス（naming service）、プリンタ１２７、記憶装置１２５、他のコンピュータ１０１、又は遠隔のホスト１３１などの、遠隔サーバ１２９にアクセスする。適切なコンピュータには、米国カリフォルニア州マウンテン・ビューのサン・マイクロシステムズ社の製造によるSPARCstation（登録商標）コンピュータが含まれる。Sun及びSolarisは、米国及び他の諸国において、サン・マイクロシステムズ社の登録商標である。すべてのSPARC商標は、米国及び他の諸国において、ライセンスの下に用いられ、スパーク・インターナショナル社の商標及び登録商標である。SPARCの商標を有する製品は、サン・マイクロシステムズ社によって開発されたアーキテクチャに基づく。他の任意の汎用コンピュータは、本発明と共に用いられるように構成できる。それぞれのコンピュータ１０１は、サン・マイクロシステムズ社のSolaris（登録商標）などの汎用オペレーティング・システムを、ネクスト・コンピュータ社からのOpenStep（登録商標）ウィンドウイング環境と共に実行する。典型的には、それぞれのコンピュータ１０１は、単一の従業員の専用であるが、コンピュータ１０１

は、それぞれ自身のコンピュータ１０１に対して、クライアントに対するサーバとして機能する、種々の異なるアプリケーションにアクセスする複数の従業員をサポートすることもできる。

【００５４】それぞれのコンピュータ１０１のアドレス指定可能なメモリ１０５は、更に、本発明の実施例のための遭遇メカニズムを実現するのに有用なソフトウェア・コンポーネントを含む。図７は、遭遇メカニズムのアーキテクチャのオブジェクト・モデルを図解している。

【００５５】遭遇メカニズムでは、それぞれの従業員又はユーザは、パーソン・オブジェクト（person object）１３７を有する。パーソン・オブジェクト１３７の参照は、通常はユーザ・ハンドルを作成するのに用いられている、ユーザ・ネームやアカウントＩＤなどのユーザに特有の情報を、記憶する。パーソン・オブジェクト１３７は、更に、ビットマップされたイメージ、ポストスクリプト・データ、ドロー・オブジェクト（draw object）、又はテキスト・ストリング（列）などのエンカウンタ・ウィンドウに表示されるユーザの所望の表現を記憶する。これらの表現は、自動的に又はマニュアルに作成され、ユーザによって修正される。多くの異なるコンピュータ１０１に亘って分散する多数のユーザを有する好適実施例では、それぞれの従業員のエンカウンタ・ウィンドウ１４３を周期的に更新することの専用の大量のネットワーク通信が存在する。従って、パーソン・オブジェクト１３７は、好ましくは、コンパクトなデータ構造を有し、遭遇メカニズムの種々のコンポーネントの間を移動する情報の量を減少させる。パーソン・オブジェクト１３７の参照は、好ましくは、ユーザの名前を与えるキューに応答して、与えられたパーソン・オブジェクト１３７にハンドルを提供するネーミング・サービスなどの、中央化されたデータベース１２９に記憶される。

【００５６】更に、パーソン・オブジェクト１３７の選択されたデータは、ビジネス・カード・オブジェクト１３９を介して見ることができる。好適実施例では、ビジネス・カード１３９は、エンカウンタ・ウィンドウ１４３又はパーソン・オブジェクト１３７からの他のクライアント・アプリケーションによって生成されるパーソン・オブジェクト１３７上のビュー（view）である。ビジネス・カード１３９は、それぞれのユーザに対して、ユーザのフルネーム、住所、電子メール・アドレス、電話番号、ファクシミリ番号などの、ユーザにコンタクトするために有用な情報を表示する。好適実施例では、ビジネス・カード１３９は、拡張可能であり、それぞれのユーザが、ローカルにキャッシュされたビジネス・カード１３９のコンポーネントに、他の従業員に関する追加的な情報を記憶するための１又は複数の新たなフィールドを定義することを可能にする。このような情報は、個人に関する静的なデータ（例えば、ページャの数など）か、又は、交流を容易にする機能的な情報（例えば、ページャをダイアルするサービスの参照）かのどちらかである。

【００５７】ビジネス・カード・オブジェクト１３９は、更に、表された従業員との通信（コミュニケーション）を開始する方法を含む。ビジネス・カード・オブジェクト１３９は、電子メール・サービスやテレビ会議などの既存のデスクトップ通信機能を統合している。一般に、分散コンピューティング環境では、ユーザのローカルな通信クライアントは、別の従業員の通信サーバのオブジェクト参照を、オブジェクト・リクエスト・ブローカに送り、次に、これが、通信サーバのハンドルをクライアントに戻す。次に、ローカルなクライアントが、通信サーバとの通信を直接に開始する。他のメカニズムも可能である。例えば、電子メール通信のためには、カレント・ワーカは、別の従業員の電子メール・アドレスを、その従業員のパーソン・オブジェクト１３７から要求し、それを、メール・ツールに送り、メール・ツールが、受け手への電子メール・メッセージを作成する。

【００５８】やはり好適実施例において、それぞれのパーソン・オブジェクト１３７又はビジネス・カード１３９は、これらのオブジェクトがアプリケーションの間でカットされペーストされることを可能にするペーストボード・タイプであり、それによって、ユーザは、従業員に関する情報を、通信サービスを介して、それぞれの間で送ることができる。

【００５９】更に、それぞれのコンピュータ１０１においては、単一のエンカウンタ・ウィンドウ１４３が提供される。それぞれのエンカウンタ・ウィンドウ１４３は、種々のパーソン・オブジェクト１３７に関連するアイコン２２などの表現を表示する。複数のコンピュータ１０１に亘って、複数のエンカウンタ・ウィンドウ１４３が存在し、従って、単一のパーソン・オブジェクト１３７の表現が、任意の数のこれらのウィンドウ１４３に現れる。エンカウンタ・ウィンドウ１４３は、意識のモード、通信機能、その中の従業員の表現などの、上述した機能を提供する。

【００６０】他の従業員の意識と通信とが望ましいが、従業員は、プライバシの感覚を維持し強化するために誰がその従業員にアクセスを有するのかを制御できなければならない。この実施例では、本発明は、従業員の利用可能性の顕著な合図のための十分なメカニズムを提供し、それによって、他の従業員は、個人のプライバシを保護する社会的なメカニズムに依拠することができる。更に、特定のメカニズムがエンカウンタ・ウィンドウにおいて与えられ、他のどの従業員がカレント・ワーカへのアクセスを有するのかを制御する。この従業員は、テレビ会議などの通信サービスの中の種々のものをイネーブル又はディセーブルして、他の従業員によってコンタクトされる程度を制御できる。更に、ユーザは、意識の

レベルを、開いた、極小の、又は閉じたモードのどれかを、好ましくは、モード・アイコン１８を用いて選択することにより、制御することができる。

【００６１】それぞれのパーソン・オブジェクト１３７は、１又は複数の遭遇を意識しているアプリケーション１３１に間接的に知られている。一般に、それぞれの遭遇を意識しているアプリケーション１３１は、一度にひとりの従業員によって用いられる。しかし、別の実施例では、遭遇を意識しているアプリケーション１３１は、他のコンピュータ１０１からの複数のユーザによってアクセスされ、アプリケーション１３１が遠隔のクライアントに対するサーバになり得る。

【００６２】任意の時点で、アプリケーション１３１のユーザは、特定の機能と特定のデータとにアクセスしている。この情報は、ファイル・ネーム、オブジェクト・ネーム、ポインタ、又はその他の手段によって表される。ユーザが用いられている機能又はデータを変更する度に、遭遇を意識しているアプリケーション１３１は、状態メッセージを、その遭遇プロキシ・オブジェクト（proxy object）１３５に送る。状態メッセージは、その従業員が現に用いているデータ又は機能、あるいは、これらの組合せを特定する。状態メッセージは、データ・ファイルのネームと機能の言及とを含むストリングであり、それぞれのユーザの位置を決定するのに有用な追加的な情報を含みことができる。

【００６３】それぞれの遭遇を意識しているアプリケーション１３１には、遭遇プロキシ・オブジェクト１３５が関連している。遭遇プロキシ・オブジェクト１３５は、遭遇を意識しているアプリケーション１３１に、遭遇サーバ１４１への通信メカニズムを提供する。遭遇プロキシ・オブジェクト１３５は、アプリケーション１３１から、アプリケーション１３１におけるユーザの現在の位置を記述する状態メッセージを受け取る。遭遇プロキシ・オブジェクト１３５は、従業員のパーソン・オブジェクト１３７へのハンドルを取得することができる。遭遇プロキシ・オブジェクト１３５は、また、それ自身が関連しているアプリケーション１３１のアイデンティティ（ＩＤ）を知っている。遭遇プロキシ・オブジェクト１３５は、状態メッセージを用いて、従業員のハンドルとアプリケーションのアイデンティティとを拘束（バインド）し、それを、遭遇サーバ１４１に送る。このようにして、アプリケーション１３１は、遭遇サーバ１４１と直接にインターフェースする必要がない。

【００６４】遭遇プロキシ・オブジェクト１３５は、更に、遭遇サーバ１４１に、遭遇を意識しているアプリケーション１３１に含まれる一致オブジェクト１３３へのハンドルを提供する。これによって、サーバ１４１は、一致オブジェクト１３３と直接に通信して、それに、２人のユーザのタスク近接性の判断のためにユーザの位置データを送ることが可能になる。一致オブジェクト１３３へのハンドルは、好ましくは、アプリケーション１３１が新たなプロセスとして実行されるときに、遭遇サーバ１４１に提供される。

【００６５】それぞれのコンピュータ１０１上のそれぞれのユーザに対して、遭遇サーバ１４１が提供される。遭遇サーバ１４１は、エンカウンタ・ウィンドウ１４３に、パーソン・オブジェクト１３７の表現が更新されなければならないときに、与えられたユーザを知らせる。エンカウンタ・ウィンドウ１４３は、これを、同じコンピュータ１０１上の遭遇プロキシ・オブジェクト１３５と他のコンピュータ１０１上の遭遇プロキシ・オブジェクト１３５とから受け取った状態メッセージに従って行う。ユーザが２つのコンピュータ１０１にログオンしている場合には、そのユーザには、２つの遭遇サーバ１４１が存在する。同様にして、複数のユーザに対する単一のコンピュータには、複数のサーバ１４１が存在しうる。

【００６６】遭遇サーバ１４１は、コンピュータ１０１上の遭遇を意識しているアプリケーション１３１のリストを維持して、それぞれのアプリケーション１３１に関する情報を、その遭遇プロキシ・オブジェクト１３５から受け取る。サーバ１４１は、更に、そのユーザに対してどのアプリケーション１３１が現にアクティブであるか（他のアプリケーション１３１が、背後で動作していることもあり得る）を識別する情報を保持する。この情報は、ウィンドウイング環境によって提供され、アクティブなアプリケーション１３１が変更する際に、時々、更新される。

【００６７】遭遇サーバ１４１は、そのコンピュータ１０１上の遭遇プロキシ・オブジェクト１３５から、及び、他の遭遇サーバ１４１から、状態メッセージを受け取り、これらの状態メッセージ１４７を記憶する。遭遇プロキシ・オブジェクト１３５からの状態メッセージは、アプリケーション１３１と、アプリケーション１３１におけるユーザの位置と、ユーザのハンドル及び位置と、もし存在すれば、アプリケーション１３１に含まれる一致オブジェクト１３３へのハンドルと、を識別する。好適実施例においては、遭遇サーバ１４１は、遭遇プロキシ・オブジェクト１３５からの状態メッセージを受け取るときに、それへのタイムスタンプを加える。タイムスタンプは、エンカウンタ・ウィンドウ１４３に現れるアイコン２２の順序を決め、タスク近接性を判断するのに有用である。遭遇サーバ１４１は、次に、状態メッセージをネットワーク１２３上のすべての他のサーバ１４１に送る。

【００６８】遭遇サーバ１４１は、状態メッセージを受け取ると、受け取ったメッセージを、記憶されていた状態メッセージ１４７と比較して、同じアプリケーション・タイプ又はアプリケーション・ネームを含む状態メッセージ、又は、一致基準を識別して、そのような状態メ

ッセージに含まれる位置データを一致オブジェクト１３３に送り、状態メッセージにおいて特定されるユーザがその位置に従ってタスク近接であるかどうかを判断する。遭遇サーバ１４１は、好ましくは、現にアクティブであるアプリケーション１３１の一致オブジェクト１３３を生じさせる。

【００６９】好適実施例では、タスク近接性の判断は、一致オブジェクト１３３によって行われる。それぞれのアプリケーション１３１は、アプリケーションのタイプに特定のタスク近接性規則を適用する一致オブジェクト１３３を有する。更に、遭遇サーバ１４１は、また、一致オブジェクト１３３のそれ自身のライブラリを有し、それぞれが、異なるタイプのアプリケーション１３１に対してタスク近接性を判断するように構成されている。

【００７０】アプリケーション１３１は、一致オブジェクト１３３を有しない場合には、遭遇サーバ１４１に対して、ファイル・ネームの比較や、ファイル・ネーム及びアプリケーション・タイプや、ファイル・ネーム及びタイム・スタンプや、更には、データの他の任意の組合せなどの、ライブラリから用いるための一致オブジェクト１３３のタイプを示す。アプリケーション１３１が用いるべき一致オブジェクト１３３のタイプを示さない場合には、遭遇サーバ１４１は、好ましくは、単純なストリングの比較を行う一致オブジェクト１３３を用いる。

【００７１】アプリケーション１３１が一致オブジェクト１３３を有する場合には、アプリケーション１３１は、好ましくは、遭遇プロキシ・オブジェクト１３５を介して、遭遇サーバ１４１を用いて、一致オブジェクトを登録する。遭遇サーバ１４１は、ハンドルを受け取り、一致オブジェクト１３３に記憶する。遭遇サーバ１４１は、タスク近接性の判断を望むときには、ハンドルを用いて、現にアクティブなアプリケーション１３１の一致オブジェクト１３３を生じさせる。

【００７２】別の実施例では、遭遇を意識しているアプリケーション１３１が開始する際には、その一致オブジェクト１３３は、遭遇サーバ１４１にコピーされ、それによって、一致オブジェクト１３３を遭遇サーバ１４１に移動させる。この実施例は、比較的高速の実現をサポートする。

【００７３】一致オブジェクト１３３は、サーバから、２つの状態メッセージを受け取るが、一方は、ユーザの現在の位置を記述し、もう一方は、典型的には第２のコンピュータ１０１上で作業している別のユーザの位置を記述する。一致オブジェクト１３３は、所定のタスク近接性関数を用いて、ユーザが相互にタスク近接であるかどうかを判断する。タスク近接性関数は、それぞれのユーザに対する位置データの間のストリングの比較か、あるいは、タイムスタンプ、データ・ファイルなどの算術的すなわちブール代数的な比較を含む、より複雑な関数である。

【００７４】一致オブジェクト１３３は、２人のユーザがタスク近接であるかどうかを示す、あるいは、既に述べたタスク近接性の複数の可能なレベルの中の１つを示す適切な状態値を戻す。一致オブジェクト１３３からの結果により、遭遇サーバ１４１が、そのエンカウンタ・ウィンドウ１４３に知らせる。次に、エンカウンタ・ウィンドウ１４３がそのディスプレイを更新し、タスク近接な従業員に対して新たな表現を追加するか、既存の１つを除去するか、又は、全く何の変更もしないか、のいずれかである。

【００７５】それぞれのコンピュータ１０１上には、活動モニタ１４５が提供されている。活動モニタ１４５は、キーボード、マウス、及び他の入力装置をモニタして、従業員が現にコンピュータ１０１を用いているかどうかを判断する。所定の長さの時間の間に活動が存在しない場合には、活動モニタ１４５は、遭遇サーバ１４１に、そのユーザは暇（アイドル状態）であるとのメッセージを送る。この状態メッセージは、ユーザのパーソン・オブジェクトへのハンドルと、ユーザのマシン識別番号と、アクティブ／アイドル・フラグとを含む。遭遇サーバ１４１は、次に、状態メッセージを、ネットワーク１２３上のすべての他の遭遇サーバ１４１に送り、ユーザがアクティブではないことを示す。残りの遭遇サーバ１４１は、それに従って、エンカウンタ・ウィンドウ１４３を更新する。別の従業員が、アイドル状態にある従業員のビジネス・カード１３９を持ち出す際には、それは、従業員がアクティブではないことを示す。ユーザが再びアクティブになるときには、活動モニタ１４５は、アクティブ・フラグを有する状態メッセージを、その遭遇サーバ１４１に送り、これが再び、状態メッセージをネットワーク１２３に放送する。

【００７６】遭遇メカニズムの動作の例が、図８に図解されている。別個の遠隔コンピュータ上で作業している又は同じコンピュータの上で作業している２人の従業員ＡおよびＢを、考える。それぞれの従業員は、遭遇を意識しているアプリケーションを有するものとし、この場合には、エディタ１３１ａ及び１３１ｂとし、それぞれが、遭遇プロキシ・オブジェクト１３５と一致オブジェクト１３３とを含む。それぞれのアプリケーション１３１は、好ましくは、アプリケーション１３１がアクティブになるときに通知を提供するアプリケーション・フレームワークを用いて作られている。それぞれの従業員は、自分自身の遭遇サーバ１４１ａ及び１４１ｂと、エンカウンタ・ウィンドウ１４３ａ及び１４３ｂとを有する。それぞれの遭遇サーバ１４１は、サーバ１４１に関連するユーザによって用いられているアプリケーション１３１の一致オブジェクト１３３へのハンドルを有する。複数のコンピュータ１０１の間のすべての通信は、遭遇サーバ１４１を介してのものである。それぞれの従業員は、自分自身のエンカウンタ・ウィンドウ１４３ａ

及び１４３ｂを開いたモードにしていると想定される。この例では、一致オブジェクト１３３は、遭遇を意識しているアプリケーション１３１に関連しており、上述したことであるが、遭遇サーバ１４１と関連している。また、活動モニタ１４５は、説明を簡単にするために、示されていない。

【００７７】従業員Ｂは、データ・ファイルＹの上で、エディタである自分のアプリケーション１３１ｂにおいて作業している８００。従業員Ａは、やはりエディタであるアプリケーション１３１ａを始動させて８０１、自分のコンピュータ上でアクティブなアプリケーションにする。エディタ１３１ａは、アプリケーション・フレームワークを介して、遭遇プロキシ・オブジェクト１３５ａに、エディタ１３１ａがアクティブであることを告げる８０３。遭遇プロキシ・オブジェクト１３５ａは、状態メッセージを、遭遇サーバ１４１ａに送り８０７、エディタ１３１ａがアクティブであることを示す。状態メッセージは、エディタ１３１ａの名前と、ゼロの位置と、エディタ１３１ａに関連する一致オブジェクト１３３ａへのハンドルとを含む。この位置は、従業員Ａは編集するデータ・ファイルをまだ選択していないので、当初は、ゼロ（ヌル）である。位置がゼロであることは、従業員Ａが、その時点では他のどの従業員ともタスク近接ではあり得ず、従って、従業員Ａは、どの他の従業員のエンカウンタ・ウィンドウ１４３にも現れず、また、他のどの従業員も従業員Ａのエンカウンタ・ウィンドウ１４３ａに現れないことを示している。一致オブジェクト１３３へのハンドルは、サーバ１４１ａを有するオブジェクトを登録するのに用いられる。

【００７８】遭遇サーバ１４１ａは、エディタの名前、一致オブジェクト１３３ａ、及び従業員Ａの位置を含むエディタ１３１に対するハンドルを、状態メッセージ８１２ａとして記憶する８０９。ひとりの従業員は、アクティブなアプリケーションは、一度に１つしかもてないので、この情報は、従業員Ａの現在のアプリケーション及び位置を記述している。従業員Ａの遭遇サーバ１４１ａは、メッセージを、エンカウンタ・ウィンドウ１４３ａに送り８１１、従業員Ａの現在の位置を示す。位置がゼロであるので、他の従業員のタスク近接性を判断するために一致オブジェクトを生じさせる必要はなく、従って、エンカウンタ・ウィンドウ１４３ａは、すべての現在のアイコン２２又は他の従業員をクリアする。一致オブジェクト１３３ａのハンドルを記憶することによって、１４１ａは、結果的に、そのオブジェクトを生じさせて、従業員Ａとそれ以外の従業員との間のタスク近接性を判断する。

【００７９】遭遇サーバ１４１ａは、また、ネットワーク上のすべての他の遭遇サーバ１４１に状態メッセージを、マルチキャストする８１３。この状態メッセージは、従業員Ａのパーソン・オブジェクト１３７ａへのハンドルと、従業員Ａのコンピュータである遭遇サーバ１４１ａを実行するコンピュータのための識別番号と、遭遇サーバ１４１ａによって生成されたタイムスタンプと、エディタ１３１ａの名前と、ゼロである従業員Ａの位置と、を含む。

【００８０】従業員Ｂの遭遇サーバ１４１ｂはこの状態メッセージを受け取る。この遭遇サーバ１４１ｂは、その記憶された状態メッセージ８１２ｂをサーチして８１５、従業員Ａのハンドル、及び、先に受け取った状態メッセージへの従業員Ａのマシン識別番号と一致するように試みる。これらの記憶されたメッセージは、従業員Ｂが位置Ｙにいることを示す状態メッセージを含む。先に記憶された状態メッセージが存在する場合には、これは、従業員Ａは、過去のある時点で従業員Ｂとタスク近接であったことを示す。従って、遭遇サーバ１４１ｂは、更に、エンカウンタ・ウィンドウ１４３ｂが、アイコン又はそれ以外のイメージなどの従業員Ａの表現を表示しているかどうかを判断する。そうであれば、遭遇サーバ１４１ｂは、メッセージをエンカウンタ・ウィンドウ１４３ｂに送り、従業員Ａの表現を除去するが、これは、従業員Ａのゼロの位置のためであり、従業員Ａは、従業員Ｂが何をしているかとは関係なく、もはや従業員Ｂとはタスク近接でない。先に記憶されたメッセージが存在しなかった場合には、エンカウンタ・ウィンドウ１４３ｂは、更新される必要がない。

【００８１】エンカウンタ・ウィンドウ１４３ｂは、従業員Ａに対するメッセージを記憶し８１９、従業員Ａに関する任意の先の状態メッセージに代える。従業員Ａの現在の位置はゼロであるから、従業員Ａが従業員Ｂに対してタスク近接であるかどうかを判断する必要がない。

【００８２】従業員Ａは、次に、データ・ファイルＸをロードして、その編集を開始する。従業員Ａのエディタ１３１ａは、メッセージをその遭遇プロキシ・オブジェクト１３５に送り８０３、位置の変化を示す。この例においては、エディタ１３１は、所定の時間周期を用い、データ・ファイルの一致（ＩＤ）に基づいて、タスク近接性を判断する。ここでは、エディタ１３１の内部での機能の一致は、要求されない。この例では、次に、エディタ１３１ａからの遭遇プロキシ・オブジェクト１３５ａへのメッセージは、データ・ファイルＸに基づく新たな位置を示し、この位置は、「位置Ｘ」と称される。他の実施例では、エディタ１３１又は他のアプリケーションが用いている機能は、この位置に含まれる。

【００８３】遭遇プロキシ・オブジェクト１３５ａは、現在の位置Ｘを有する状態メッセージとエディタ１３１ａの名前とを遭遇サーバ１４１ａに送る８０７。遭遇サーバ１４１ａは、その記憶されたメッセージ８１２ａをサーチして８０５、より限定されたタスク近接機能のために、エディタ１３１ａと同一のアプリケーション・タイプ又はアプリケーション・ネームを示すメッセージ

が、存在するかどうかを判断する。同じアプリケーション・タイプ／ネームを有する状態メッセージが見いだされた場合には、遭遇サーバ１４１ａは、一致オブジェクト１３３ａを生じさせて８２１、見い出された状態メッセージに関連する従業員の位置が従業員Ａの位置である位置Ｘと同じであるかどうかを判断する。遭遇サーバ１４１ａは、見い出された状態メッセージから取られた他の従業員の位置に送られる。

【００８４】この場合には、遭遇サーバ１４１ａは、（先に受け取られた）状態メッセージを見つけるが、このメッセージは、従業員Ｂもまたエディタ１３１ｂを用いているが、データ・ファイルＹと共にであり、従って、位置Ｙにおいてであることを示している。再び、これらの複数のエディタは、同じネームと同じタイプを有することになる。一致オブジェクト１３３ａは、従業員Ａの現在の位置である位置Ｘにアクセスするが、この理由は、これが、エディタ１３１ａに関連しているからである。一致オブジェクト１３１ａは、位置Ｘと位置Ｙとが同じではないことを判断し、タスク近接のフラグをそのように指示するように戻す８２３。従って、従業員Ｂは、この時点では、エンカウンタ・ウィンドウ１４３ａには現れず、従業員Ａもエンカウンタ・ウィンドウ１４３ｂには現れない。

【００８５】遭遇サーバ１４１ａは、従業員Ａの現在の位置を示す状態メッセージを送る８１３。再び、この状態メッセージは、従業員Ａのパーソン・オブジェクト１３７ａへのハンドルと、従業員のコンピュータの識別番号と、タイムスタンプと、エディタ１４１ａのネームと、位置Ｘとを含む。遭遇サーバ１３１ｂがこのメッセージを受け取り、従業員Ａに対する先の状態メッセージを、現在のものと代える。従業員Ａに対する先の位置は、ゼロであるから、エンカウンタ・ウィンドウ１４３ａを更新する必要はない。

【００８６】遭遇サーバ１４１ｂは、新たな状態メッセージ及びその中で特定されるアプリケーション・タイプ／ネームを、現在のアプリケーション・タイプ／ネーム、ここではエディタ１３１ｂと比較する。これらの一致と遭遇サーバ１４１ｂとは、エディタ１３１ｂに関連する一致オブジェクト１３３ｂを生じさせ、受け取った状態メッセージから位置Ｘに送る。一致オブジェクト１３３ｂは、位置Ｘ及び位置Ｙの位置を比較し、一致がないことを判断し、そのように、遭遇サーバ１４１ｂに告知する。従って、従業員Ｂのエンカウンタ・ウィンドウ１４３ｂを更新する必要はない。

【００８７】従業員Ａは、次に、データ・ファイルＹをロードする。再び、エディタ１３１ａは、その遭遇プロキシ・オブジェクト１３５ａに、新たな位置である位置Ｙを知らせ、遭遇プロキシ・オブジェクト１３５ａは、位置情報を有する状態メッセージを、遭遇サーバ１４１ａに送る。遭遇サーバ１４１ａがその記憶されたメッセージをサーチする８０５と、遭遇サーバ１４１ａは、従業員Ｂもまたエディタ１３１を用いていることを示すメッセージを識別する。遭遇サーバ１４１ａは、一致オブジェクト１３３ａを生じ８２１、これにより、従業員Ｂの位置Ｙが従業員Ａの位置Ｙと一致することが判断され、タスク近接フラグをこの効果に戻す。遭遇サーバ１４１ａは、従業員Ｂのパーソン・オブジェクト・クライアントへのハンドルと表現（アイコンなど）とを有するメッセージを、エンカウンタ・ウィンドウ１４３ａに送り８１１、このエンカウンタ・ウィンドウ１４３ａが、表現を表示する。従業員Ａは、これで、従業員Ｂが自分に対してタスク近接であることがわかる。状態メッセージに含まれるタイムスタンプは、エンカウンタ・ウィンドウ１４３ａによって用いられ、従業員Ｂ又はそれ以外の者の表現の現れの順序を付ける。

【００８８】遭遇サーバ１４１ａは、従業員Ａの現在の位置Ｙを有する状態メッセージを、他の遭遇サーバ１４１に、送る８１３。遭遇サーバ１４１ｂは、このメッセージを受け取り、それを、先に記憶された、メッセージと比較して８１５、このメッセージを、従業員Ｂの位置を特定する先の状態メッセージからの位置と共に、一致オブジェクト１３３ｂに送る。この一致オブジェクト１３３ｂは、また、従業員Ａ及びＢの位置を、タスク近接であるとして、一致させる。遭遇サーバ１４１ｂは、従業員Ａのパーソン・オブジェクト・クライアント１３７ａへのハンドルとアイコンと共に、メッセージをエンカウンタ・ウィンドウ１４３ｂに送り、従業員Ｂのエンカウンタ・ウィンドウ１４３ｂは、従業員Ａの表現を用いて、それ自身を更新する。

【００８９】従業員Ａが従業員Ｂと交流することを望む場合には、従業員Ａは、エンカウンタ・ウィンドウ１４３上のボタンの中の１つ又は従業員Ｂのビジネス・カードを用いて、テレビ会議などの通信メカニズムを開始させることができる。遭遇サーバ１４１ａは、状態メッセージの中に、従業員Ｂへのオブジェクト参照と、従業員Ｂのコンピュータのマシン識別番号とを記憶する。要求されたときには、サーバ１４１ａは、これらの値を用いて、従業員Ｂのコンピュータの上の通信サービスに接続する。

【００９０】ユーザがそのエンカウンタ・ウィンドウ１４３を閉じたモードにおく場合には、その遭遇サーバ１４１は、状態メッセージを継続して受け取り８１３、記憶する。これによって、遭遇サーバ１４１が、そのメカニズムの現在の状態を維持することが可能になり、それによって、ユーザがそのエンカウンタ・ウィンドウ１４３を極小又は開いたモードに設定する場合であれば、エンカウンタ・ウィンドウ１４３を更新する。更に、閉じたモードにある場合には、エンカウンタ・ウィンドウ１４３は、ユーザに対してゼロの位置を有する状態メッセージを送出し、それによって、ユーザは、どの他のユー

ザに対しても、もはや、タスク近接ではなくなる。従って、他のユーザのエンカウンタ・ウィンドウ１４３が、更新される。

【００９１】ユーザがエンカウンタ・ウィンドウ１４３を極小モードにおく場合には、遭遇サーバ１４１は、それが送出する状態メッセージにフラグを追加し、その従業員に対する極小モードを指示する。他の遭遇サーバ１４１が状態メッセージにおいてこのフラグを受け取る際には、それぞれの遭遇サーバ１４１は、その関連するエンカウンタ・ウィンドウ１４３に、特定のユーザに対するこのフラグを知らせる。それぞれのエンカウンタ・ウィンドウ１４３は、次に、シルエット４２などの極小モードの適切な表現を用いて、そのディスプレイを更新する。

【００９２】図７を参照すると、好適実施例においては、パーソン・オブジェクト１３７は、２つの部分から形成される。すなわち、ライトウェイト・クライアント・コンポーネントと、よりヘビーウェイトのサーバ・コンポーネントとである。クライアント・コンポーネントは、アドレスを、サーバ・コンポーネントに提供し、ユーザの表現をキャッシュするなどパフォーマンスを最適化するために種々の動作を行い、エンカウンタ・ウィンドウ１４３を更新し、通信サービスに接続するために必要な時間を減少させる。パーソン・オブジェクト１３７を送ることは、一般に、サーバ・コンポーネントのアドレスを送ることから構成される。

【００９３】パーソン・オブジェクト・クライアントは、ユーザ、遭遇を意識しているアプリケーション、及びエンカウンタ・ウィンドウ１４３の間で通信される。パーソン・オブジェクト・クライアントの値、例えばネームが求められる場合には、クライアントは、パーソン・オブジェクト１３７サーバにアクセスして、次に、値をクライアントに戻し、クライアントがそれを要求側のオブジェクトに与える。従って、エンカウンタ・ウィンドウ１４３は、それ自身を更新している時には、ローカルなパーソン・オブジェクト１３７クライアントにおいて利用可能な表現が既にない場合には、パーソン・オブジェクト１３７のサーバ・コンポーネントから、表現のためのデータを取得する。好適な構成では、パーソン・

オブジェクト１３７クライアントは、不変の（あるいは、ほとんど変化しない）値、例えば、ユーザの名前やアイコン表現などを、キャッシュすることができる。プロセスや振る舞いを実行することに依存する情報、例えば、状況フラグ又は現在の活動レベルは、キャッシュされない。パーソン・オブジェクト１３７クライアントは、好ましくは、他のオブジェクトに対しては、読み出し動作だけを提供し、そのようなオブジェクトが値を更新することを許容しない。パーソン・オブジェクト１３７サーバは、それが表している人間によって制御され、この人間は、適切なインターフェースを通じて、サーバ内の値を更新することが許される。

【００９４】以上の説明は、例示であり、限定的なものではない。本発明の多くの変更が、この開示に基づけば、当業者には明らかであろう。本発明の範囲は、従って、その均等物の完全な範囲と共に、冒頭の特許請求の範囲を参照して判断されるべきであり、上記の説明には制約されない。

【図面の簡単な説明】

【図１】本発明の実施例による、開いたモードのコンパクトなフォーマットでの、コンピュータ・ディスプレイ上のエンカウンタ・ウィンドウの図解である。

【図２】タスク空間の図解である。

【図３】開いたモード・フォーマットでのエンカウンタ・ウィンドウの図解である。

【図４】開いたモードの拡張されたフォーマットでのサーチ及びソート・メカニズムを提供するエンカウンタ・ウィンドウの別の実施例の図解である。

【図５】図５ａ及び図５ｂ共に、極小モードにおけるエンカウンタ・ウィンドウの図解である。

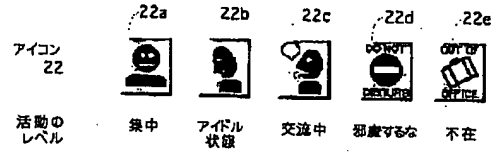
【図６】本発明をサポートするコンピュータ・システムのハードウェア要素のブロック図である。

【図７】本発明のある側面のためのシステム・アーキテクチャのオブジェクト・モデルである。

【図８】エンカウンタ・メカニズムの典型的な振る舞いを図解するイベント・トレースである。

【図９】種々のレベルの利用可能性を図解するアイコンの組である。

【図９】



【図1】

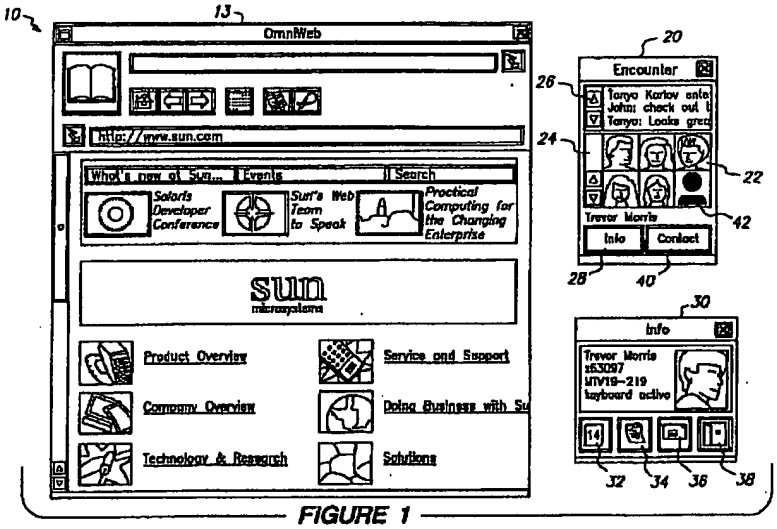
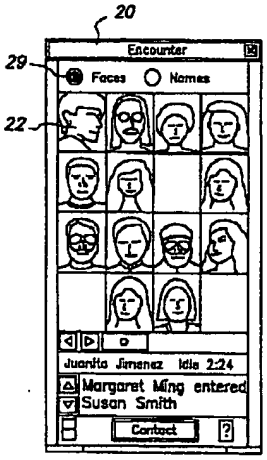
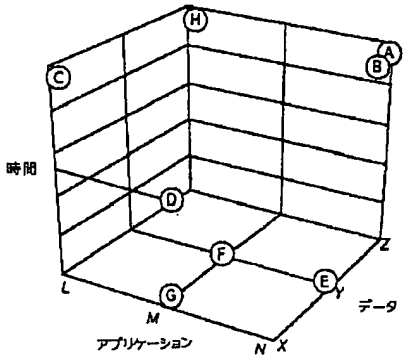


FIGURE 1

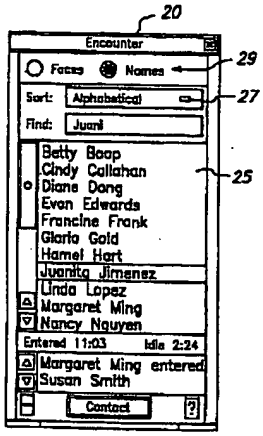
【図3】



【図2】



【図4】



【図5】

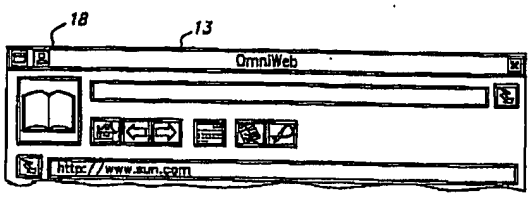


FIGURE 5a

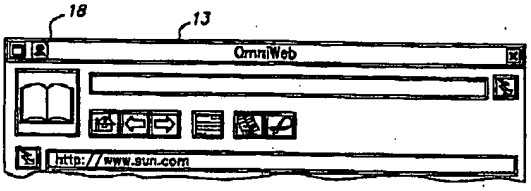
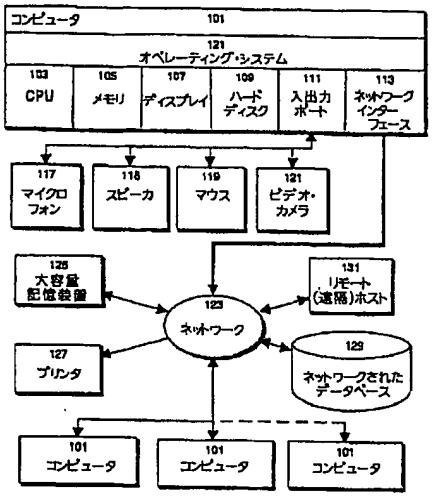
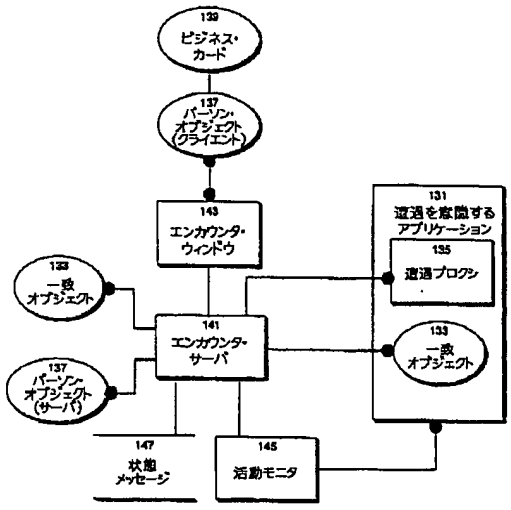


FIGURE 5b

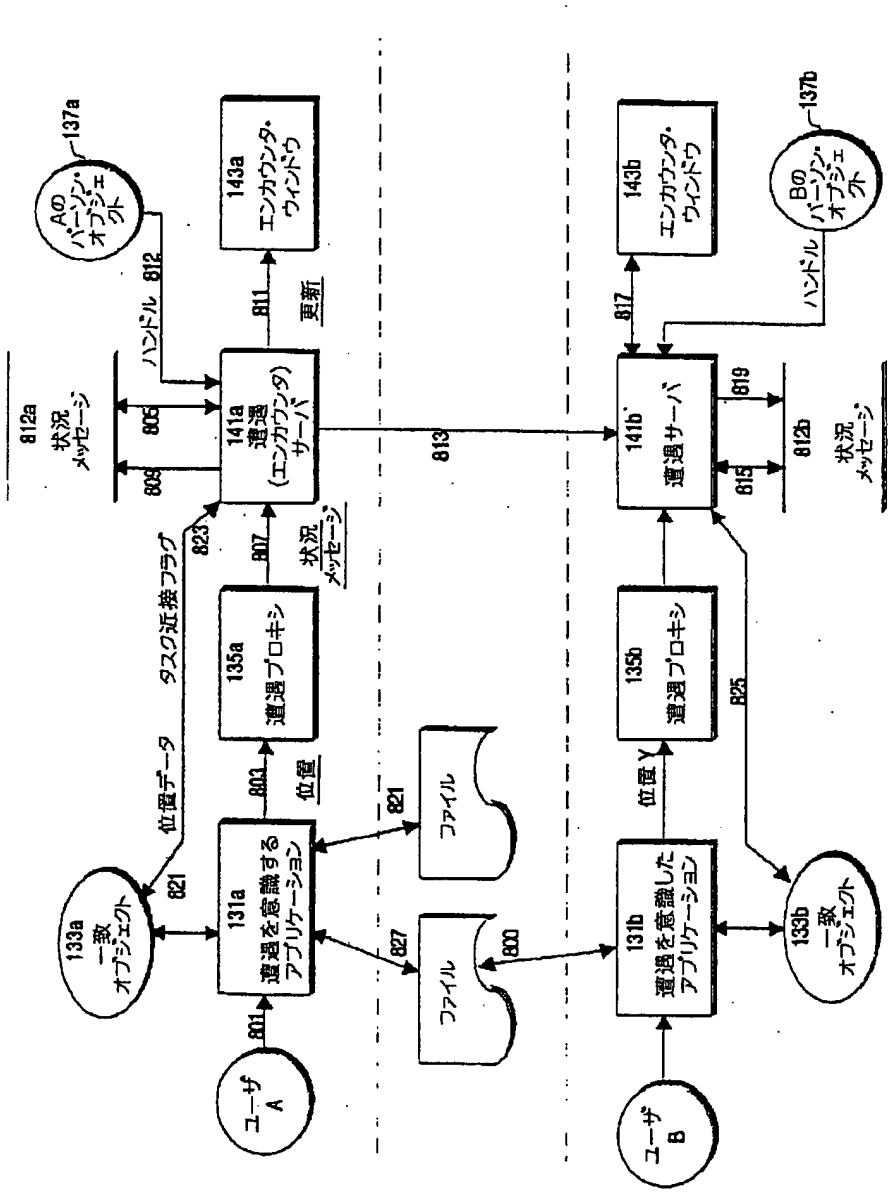
【図6】



【図7】



【図８】



【手続補正書】
【提出日】平成９年４月２３日
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】発明の名称
【補正方法】変更
【補正内容】
【発明の名称】　　コンピュータ業務環境において類似のタスクを行っている他人の認識を可能にするシステム及び方法

フロントページの続き

(71)出願人 597004720
2550 Garcia Avenue, MS PAL1-521, Mountain View, California 94043-1100, United States of America
(72)発明者 エレン・アイザックス
アメリカ合衆国カリフォルニア州, サニーヴェール（番地なし）
(72)発明者 トレヴァー・モーリス
アメリカ合衆国カリフォルニア州94043, マウンテン・ヴュー, シェアランド・アベニュー 181
(72)発明者 トーマス・ロドリゲス
アメリカ合衆国カリフォルニア州94110, サンフランシスコ, フォルサム・ストリート 2473
(72)発明者 アラン・ラバーグ
アメリカ合衆国カリフォルニア州94404, フォスター・シティ, エメラルド・ベイ・レーン 605
(72)発明者 リック・レヴェンソン
アメリカ合衆国カリフォルニア州94040, マウンテン・ヴュー, ライムトゥリー・レーン 1873

【外国語明細書】

Title of Invention : SYSTEM AND METHOD ENABLING AWARENESS OF OTHERS WORKING ON SIMILAR TASKS IN A COMPUTER WORK ENVIRONMENT

Claims

1. A computer system for a plurality of users and providing to a first user a visual representation of selected second users who are task proximate to the first user, comprising:

a plurality of computers, each computer having a plurality of executable applications;

one of the plurality of computers being a first computer of a first user, the first computer having a first application accessing a first datum;

each of the remaining plurality of computers being a second computer of a second user, each second computer having a second application accessing a second datum; and,

the first computer having a first user interface display displaying, for any first application, visual representations of selected second users, the second users each selected according to a first relationship between the first datum and a second datum.

2. The computer system of claim 1, wherein the first relationship between the first datum and the second datum comprises the first datum being the same as the second datum.

3. The computer system of claim 2, wherein the first relationship between the first datum and the second datum further comprises the first application accessing the first datum within a determined length of time from the second application accessing the second datum.

4. The computer system of claim 2, wherein the first relationship between the first datum and the second datum further comprises the first application accessing the first datum at a substantially same time that the second application accesses the second datum.

5. The computer system of claim 1, wherein each application has a type, each second user is further selected according to a second relationship between the type of the first application being used by the first user, and the type of a second application being used by the second user.

6. The computer system of claim 5, wherein the second relationship comprises the first application having a same application type as the second application.

7. The computer system of claim 1, wherein each second user is further selected according to the first application being used by the first user being the same as a second application being used by the second user.

8. The computer system of claim 1, wherein the first datum is a first data file stored in a first file directory, and the second datum is a second data file stored in a second file directory, and the first relationship further comprises the first file directory being the same as the second file directory.

9. The computer system of claim 1, further comprising:

for each selected second user, the second computer of the second user having a second user interface display displaying a visual representation of the first user.

10. The computer system of claim 1 wherein:

each computer includes at least one functional component adapted to compare the datum associated with each of two users, and provide in response thereto a signal indicating whether the data has the first relationship.

11. The computer system of claim 10, wherein the functional component is further adapted to compare an application type of a first application with an application type of a second application, and provide in response thereto the signal indicating whether the first and second applications are of a same type.

12. The computer system of claim 1, wherein a visual representation of a second user is added to the first user interface display at substantially the same time as when the first user becomes task proximate to the second user according to the first relationship, and wherein a visual representation of a second user is removed from the user interface display at substantially the same time that the first user is determined to no longer be task proximate to the second user according to the first relationship.

13. A computer system for a plurality of users and providing to a first user a visual representation of selected second users, comprising:

a plurality of computers, each computer having a plurality of executable applications, each application having a type;

one of the plurality of computers being a first computer of a first user, the first computer having a first application accessing a first datum at a first time;

each of the remaining plurality of computers being a second computer of a second user, each second computer having a second application accessing a second datum at a second time; and,

the first computer having a first user interface display displaying, for any first application and any first datum, visual representations of selected second users who are task proximate to the first user, where each selected second user is individually determined to be task proximate to the first user according to at least one relationship from a group comprising:

a first relationship between the first datum and the second datum;

a second relationship between a type of the first application and a type of the second application; and,

a third relationship between the first time and the second time.

14. The computer system of claim 13, wherein a first user is task proximate to a second user where in the first relationship:

the first datum is the same as the second datum.

15. The computer system of claim 13, wherein a first user is task proximate to a second user where in the second relationship:

the type of the first application is the same as the type of the second application.

16. The computer system of claim 13, wherein a first user is task proximate to a second user where in the third relationship:

the first time at which the first user uses the first application is within a predetermined length of time to a second time the second user uses the second application.

17. The computer system of claim 13, wherein the user interface display includes a first mode of operation that provides a first visual representation for all second users who are task proximate to the first user, and a second visual representation when no other users are task proximate to the first user.

18. The computer system of claim 17, wherein:

the user interface display of the first computer further includes a second mode of operation that provides only a visual representation of whether or not at least one second user is task proximate to the first user; and,

the user interface display of each second computer displays a visual representation of the first user indicating that the user interface display of the first computer is in the second mode of operation.

19. The computer system of claim 13, further comprising:

a second computer of a second user, the second computer having a user interface display including a visual representation of the first user where the first user is task proximate to the second user, and the visual representation of the first user is a function of a display mode of the user interface display of the first computer.

20. The computer system of claim 13, further comprising a communication mechanism allowing the first user to initiate a communication to a second user represented by a visual representation in the user interface display.

21. The user interface of claim 13, wherein there is provided a first aural indication each time a second user becomes task proximate to the first user, and a second aural indication each time a second user becomes no longer task proximate to the first user.

22. The computer system of claim 13, wherein each computer further comprises:

at least one application capable of providing a message specifying a position including a datum that a user is accessing at a selected time, the message provided by an active application at about the time the user changes their position; and,

at least one match object that compares a first position of a first user and a second position of a second user, and provides a signal indicating whether the first and second users are task proximate to each other, where the first position includes the first datum, the first application, and the first time, and the second position includes the second datum, the second application, and the second time.

23. The computer system of claim 22, further comprising:

a plurality of person objects, each person object uniquely associated with a user of the computer system, each person object having a visual representation capable of being displayed in the user interface displays.

24. The computer system of any of the preceding claims, wherein the visual representation of each selected second users is periodically updated to indicate a current level of availability of the second user.

25. The computer system of any of the preceding claims, wherein the updating of the visual representation of selected user replaces a first visual representation associated with a first level of availability with a second visual representation associated with a second level of availability.

26. The computer system of any of the preceding claims, wherein the first visual representation is gradually, visually transformed into the second visual representation.

27. In a computer system for multiple users, where each user has a display device having an interface display, a computer implemented method of providing an awareness of a second user who is task proximate to a first user, comprising:

determining for the first user a first position in a first application that the first user is using;

determining for the second user a second position of a second application that the second user is using;

determining whether the first position is task proximate to the second position;

responsive to the first position being task proximate to the second position, displaying for each user a visual representation of the other user in the interface display associated with each user; and

for each user, periodically updating the visual representation of the users to indicate a current level of availability of the user.

28. The computer implemented method of claim 27, wherein periodically updating of the visual representation of selected user comprises:

replacing a first visual representation associated with a first level of availability with a second visual representation associated with a second level of availability.

29. The computer implemented method of claim 27 wherein determining whether the first position is task proximate to the second position further comprises at least one of a group of comparisons including:

comparing a first datum specified in the first position with a second datum specified in the second position;

comparing a first application specified in the first position with a second application identified in the second position;

comparing a type of a first application specified in the first position with a type of a second application identified in the second position; and,

comparing a first time specified in the first position to a second time specified in the second position.

30. The computer implemented method of claim 27, where the first datum and second datum are data files in at least one directory structure, wherein comparing a first datum specified in the first position with a second datum specified in the second position further comprises:

determining whether the first data file is in a same directory level as the second data file.

31. The method of claim 27 further comprising:

responsive to a change in the position of the first user such that the position of the first user is no longer task proximate to the position of the second user, removing the visual representation of the first user from the interface display of the second user, and removing the visual representation of the second user from the interface display associated with the first user.

32. In a computer system for multiple users, a computer readable memory accessable by a first computer of a first user, the first computer having a processor and a display device, the

memory storing at least one computer program executable by the processor that provides an awareness to the first user of a second user who is task proximate to a first user, the computer program controlling the processor to:

determine for the first user a first position in a first application executing on the processor;

receive a signal indicating for a second user a second position in a second application that the second user is using;

determine whether the first position is task proximate to the second position; and,

responsive to the first position being task proximate to the second position, display a visual representation of the second user in the interface display associated of the computer of the first user.

Detailed Description of Invention

RELATED APPLICATION

This application is related to the application Serial No. , filed on , entitled SYSTEM AND METHOD PROVIDING A COMPUTER USER INTERFACE ENABLING ACCESS TO DISTRIBUTED WORKGROUP MEMBERS, which is incorporated by reference herein. Both applications are assigned to Sun Microsystems, Inc. of Mountain View, California.

BACKGROUND

Field of the Invention

The present invention relates generally to the field of computer user interface design, and more particularly to user interfaces and methods for improved user collaboration in computer-based work environments.

Description of the Background Art

In many workplaces, a significant degree of workers' productivity is based on the ability to directly interact with other workers in order to exchange information about common problems, issues, or concerns. Informal interactions are extremely important for workers who are members of a team or a workgroup sharing various aspects of a common project. Recent research has shown that while informal interactions are responsible for a significant amount of information flow in an organization, they are under-utilized with respect to how effective they are. In particular, recent investigations of information flow in organizations have found that when

people wanted to communicate information to others in the organization, they tended to communicate through the hierarchical management chain, using formal mechanisms, such as documents and presentations, even though they did not find these mechanism to be very effective. On the other hand, when workers wanted to find out information from other parts of the organization, they tended to ask other workers who they knew and respected, thus, relying on informal interactions, particularly personal, spoken communications. This research suggests that informal interactions often lead to an improved sense of community in the workplace and team cohesion, more efficient problem solving, and an increased pool of knowledge and experiences among workers. This collection of knowledge and experience improves the performance of individual workers and workgroups as a whole, and often improves morale and job satisfaction.

Physical proximity supports group work by enabling group members to enter informal, unplanned interactions. In particular, in working environments where group members share nearby offices and workspaces, there is often an awareness of which other group members are present. This enables one group member to easily contact the other group member present and initiate a dialogue with that person on some issue of concern. The awareness of others is known to be important for enabling spontaneous, and impromptu interactions that in turn facilitate workers in coordinating their actions, and creating shared understanding of common issues. The ability to informally interact in this manner provides significant benefits to the business, and the awareness workers have of other workers facilitates this ability to interact.

Group members often have many similar or overlapping tasks to perform, and common sources of information or data to use in working on various aspects of a common problem. Many of the informal interactions between group members are directed to solving common problems in these tasks. Encountering colleagues in the course of these activities often provides opportunities for informal yet effective interpersonal communication. The awareness of other workers further facilitates the sharing of issues related to common tasks or functions.

However, several trends are combining to make it hard for work groups to stay cohesive. As organizations get larger in size, members of work groups often get distributed among different buildings on a campus, or even to globally distributed geographical sites. Companies are also embracing flexible work schedules, telecommuting, and working-at-home programs. The escalating use of computers in the corporate workplace further influences these trends.

All of these trends combine to detract from the physical access that work groups traditionally shared by being in close physical proximity to each other. An increasing amount of group work is accomplished through electronically mediated mechanisms, such as networked computer systems, facsimiles, video teleconferencing and the like. While these electronic facilities can very efficiently aid the flow of raw information across physical distances that may separate group members, they do not provide the same rich sense of awareness and opportunities for interaction shared by people who work physically in the same location.

Accordingly, it is desirable to provide a computer-based mechanism that provides to distributed work group members an analogue of the sense of awareness shared by workers who are physically near each other. In a desirable collaborative computer system, applications should gracefully provide awareness of other people who are "task proximate." Workers are task proximate when they are working on the same or related data, with the same or related applications, at about the same time. A desirable mechanism further provides a way for people to initiate a conversation or other encounter with another person who is task proximate and with whom they would like to interact.

Computer implemented communication devices of many varieties are known, but they fail to provide a generalized mechanism that produces an awareness of other workers having similar tasks for any variety of tasks. Some conventional products provide very limited information targeted to a specific type of application. For example, a conventional web browser for accessing the World Wide Web (WWW) can only determine if two or more different users

are on the same web page at the same time. Both users must have the same web browser application and the web page must be provided by a web server compatible with the web browsers. This limits the awareness of others to a very small and specific community of users and computing resources. Moreover, because the implementation is designed to determine only whether there is more than one user on the same web page, such a product fails to provide a generalized system architecture that can be used with various different applications to determine whether individuals are performing related tasks on related data in various time periods of use. Further, such products do not provide a generalized user interface that indicates the relationship between the tasks different workers are performing and that facilitates communication with them.

Other conventional products rely on geographic models to simulate shared environments of a community of users. Examples of these include *M*ulti-*u*ser *d*ungeons (MUDs) and *M*UDs, *o*bject-*o*riented (MOOs). These mechanisms provide a virtual space through which users navigate, interacting with other users sharing the same room or location in the virtual space. These mechanisms are based entirely on a location-oriented model of the virtual space, and not on a task-oriented model. Further, MOOs and MUDs are designed for use as their own environment, in a sense their own application, rather than an architecture with which other applications can operate and provide task proximity information. In addition, most MOOs and MUDs are used for entertainment purposes, and specifically to meet other users.

Other conventional computer tools merely provide directed communication facilities. For example, email or video-conferencing products allow a user to directly communicate with other workers in a particular mode. However, these products provide no information about the task proximity of users. Rather, these tools are intended for a user who already knows they want to communicate with a particular person or group of persons. Thus, they do not facilitate the type of spontaneous interactions enabled by an awareness of users who are task proximate to one another.

SUMMARY OF THE INVENTION

An embodiment of the present invention provides a mechanism that enables workers using their computers to know which other workers are "nearby" in terms of the type of work they are doing, such as the data they are accessing, the application they are using, and the time when such work was performed. In one embodiment, this relationship is known as "task proximity." One worker is "task proximate" to another worker when both are accessing similar types of data, or using similar application tools within a particular time period. The relationship between the data, tools, time period may be varied to accommodate different types of data and applications. The task proximity relationships correspond to the types of interactions fostered in physical working environments where workers share information on common tasks. The invention informs workers of the others who are task proximate, facilitating the type of spontaneous interactions found in physical working environments, since a worker who is working on a computer-implemented task might benefit from the awareness that other workers are working on similar tasks. When a worker is aware of other users in this manner, there is an increased likelihood of an interaction that will support their tasks or some other related task.

Workers will typically be using a number of different computer applications to perform their job functions and goals. While different workers may be using different applications, they may nonetheless be performing the same or similar tasks. Accordingly, another aspect of the present invention provides a system and application architecture for determining task proximity between different users, operating on different machines, using different applications at different times. In addition, because the ultimate goal is to facilitate interactions between task proximate users, the architecture of the present invention provides mechanisms for efficient and simple initiation of communications between task proximate users. Finally, the present invention provides a user interface with representations of task proximate workers, enabling each worker to easily initiate various types of interactions with other workers.

DETAILED DESCRIPTION OF THE INVENTION

The Encounter Window and Task Proximity

Referring now to Figure 1, there is shown one embodiment of a user interface of a mechanism enabling awareness of other users who are task proximate. Figure 1 illustrates the screen display, commonly called the "desktop" of a particular user, called here, the current worker. The desktop is produced on the display screen of the current worker's computer. On the desktop 10, there is an application window 13 for a browser application, and one embodiment of an encounter window 20, which is the user interface portion of the encounter mechanism. The encounter window 20 provides a visual mechanism for informing the current worker which other workers are task proximate. The encounter mechanism further provides aural indication of task proximate workers. For those workers who are task proximate, the encounter mechanism provides a means of efficiently initiating an interaction with such workers.

For each worker who is task proximate to the current worker, the encounter window 20 displays an appropriate representation of the worker. In the preferred embodiment, the representation is an icon 22. The icon 22 may a bit-mapped image of the worker. In alternative embodiments, other forms of representation may be used, such as various graphic image formats, real-time video, a simple text string, or other information, depending on the level of hardware support available to each worker, the network bandwidth available, and the level of privacy each worker desires. For workers with computers including video cameras, the representation may be created by capturing a video image of themselves. For workers without video support, an icon can be selected from a set of icons or created by the worker, or a text string may be used.

The encounter window 20 is periodically updated as new workers become task proximate to the current worker, and other workers lose their task proximity. As either the current worker or other workers context switch between applications, the encounter window 20 is updated to display the icons 22 of those workers who are then task proximate to the current worker. When a

worker on a remote computer uses an application and becomes task proximate to the current worker, the encounter window 20 provides a visual/aural cue of same, with an icon 22 appearing for the other worker in the encounter window 20. When the person is no longer task proximate their icon 22 is removed. The encounter window 20 thereby provides a subtle indication to the current worker that there has been a change in the task proximity of other workers.

Task proximity may be based on any of three distinct factors: 1) the application the worker is currently using; 2) the data the worker is accessing or manipulating; and, 3) the time at which such actions occur. These factors are variously used to define a user's position in a "task space." The strictest definition of task proximity is having two workers who are using a same function of the same application on the same data file at the same time.

The definition of position and task proximity can be independently relaxed along each of the above listed factors. For example:

- the application constraint may be relaxed so that workers viewing the same data with different applications, or application types, are still task proximate. Examples include viewing the same World Wide Web page with different web browsers, or accessing the same database table with different database applications. Alternatively, the application constraint may be tightened so that even in the same application, two workers would have to be performing the same function, such using a spell checker in a word processor, compiling code in a compiler, and the like. Workers using the same application but performing different functions may be considered to be not proximate.
- the time constraint may be relaxed so that workers accessing the same data within a predetermined time period of each other may be task proximate. For example, workers accessing the same stock quotation within a quarter-hour could be task proximate to each other. Similarly, workers accessing the same web page or email message within

one hour, or the same word processing document within one day, could also be task proximate.

- the data constraint may be relaxed so that workers using the same application, but accessing different data may be task proximate. For example, workers accessing different files in the same file directory with a file browser could be task proximate. As another example, a worker viewing the calendar of a particular person for a specific date would be task proximate to other workers viewing the same person's calendar for different dates. Similarly, when viewing a calendar on a selected date to schedule a facility such as a conference room, a worker would be task proximate to other workers viewing the calendar on that date and scheduling a different conference room.

Task proximity is preferably determined with respect to an application that is currently active, or most recently active. Task proximity can be determined by the applications, or by other software components that have information about the position of the users. This information is preferably provided by the applications. Applications that can provide this information are "encounter-aware." It is preferred that applications of a given type determine task proximity in the same or similar manner. An application's type is generally based on the nature or domain of use, for example, databases, word processors, mailer, code compilers, and so on. Some applications may be considered as having more than one type. For two applications of a given type it is preferred that there is substantially the same determination of task proximity for their respective users.

Figure 2 illustrates a variety of task proximity relationships. Figure 2 represents a task space with axes defined by the three factors described above. In the task space individual workers are represented by labeled circles. Along the application axis there are three encounter-aware applications, L, M, and N. Along the data axis there are three data files, X, Y and Z. Time is continuous. Workers A and B are both using application N to access data file Z at the

same time. If N and other appllications of N's type enforce this definition of task proximity, then both A and B would see each other's representation in their encounter windows.

Workers C and H are in application L, which relaxes the data and application constraints. Application L applies a proximity definition such that two workers are task proximate if they are both using application L or, if not both using the same application, then accessing the same data. Application L does impose a strict time requirement. Thus, worker C and worker H are task proximate since they are both in application L at the same time, but accessing different data, and they would both appear in each other's encounter window. In addition, application L considers worker H to be task proximate to workers A and B, since they are accessing the same data file Z at the same time as H, but using different applications. An example of this situation would have applications L and N being different World Wide Web browsers, and data file Z being a WWW page. As another example, workers G and F are using application M at the same time on different but related data files, and would appear in each other's encounter window.

In one preferred embodiment, task proximity is symmetrical, so that if a first worker appears in a second worker's encounter window, then the second worker appears in the first worker's encounter window. This is desired in order to prevent lurking.

However, in some instances asymmetry is preferred or necessary, for example when relaxing the time constraint. For example, workers D, F, and G are using application M which employs a task proximity definition that requires the same application, but relaxes the time constraint partially, and the data constraint. In this example then, workers D and G both access data file X, but at different times (e.g. D accesses data file X some time after G does). Thus, worker G would appear in worker D's encounter window, but D would not appear in G's encounter window.

Worker E is using application N on data file Y at the same time that workers G and F are in application M. However, they do not appear in each other's encounter windows because both

applications M and N require identity of application. These are but a few of the types of task proximity relationships possible.

While the foregoing examples refer to individual applications, the definition of task proximity is preferably consistent within a given type of application, regardless of the specific implementations of each instance of the application. Further specific examples of task proximity definitions for application classes include (but are limited to):

Application Type	Task Proximity Definition
text editor/word processor	same data file.
file browser	same directory tree, or within one directory level.
email application	same message.
code development environment	code file at same level of code tree.
calendar browser	same calendar file or same date.
web browser	same web page.

The examples merely illustrate the various ways of defining task proximity for different types of applications. For any of these classes the time constraint has not been applied because it can be relaxed by varying degrees, depending on the size of the installed based, and hence the number of workers who may become task proximate in a given time period for a given data file, or other factors.

In the preferred embodiment, the encounter mechanism provides various modes of "awareness" about other workers. The modes allow the current worker to control the degree to which the encounter mechanism intrudes upon his or her experience. The mode of awareness is specified by each worker independently, and controls both ends of the encounter mechanism, namely, how the encounter mechanism appears to the current worker, and how the current worker appears in the encounter window 20 of other workers. The mode of awareness may be

specified as applying to all applications on the desktop, or may be specified in each application individually. In the preferred embodiment of the encounter mechanism, there are three modes of awareness for the encounter mechanism, including open, minimal, and closed. The mode of awareness is selected from a menu or by clicking on the mode icon 18.

The open mode of awareness is used by workers who are receptive to interacting with other workers who are task proximate. When the encounter mechanism is specified as "open," the worker sees the representations of workers who are task proximate, and these representations are updated as workers change their task proximity. When a worker sets the encounter mechanism to be in the open mode, that worker's visual representation will appear in other workers' encounter window. Further, in the open mode one may obtain additional information about such users, such as their email address, telephone number, fax number, mail address, and the like, and initiate an interaction with any number of these workers, through such means as a video conference, email message, and the like.

In addition to the visual representation of workers in the encounter window, an aural indication may be used to indicate each time a worker becomes task proximate, or loses task proximity. Different aural indications may be used for each of these events, for example, with a long beep tone when a worker becomes task proximate, and a short beep tone when a worker loses proximity. Other sound effects may also be used.

In the preferred embodiment, the addition or removal of icons 22 to or from the encounter window 20 is done by a visual transformation of the icons 22 as the workers become task proximate or lose task proximity. The visual transformation may include a fade, wipe, dissolve, pull, or other gradual visual effects that unobtrusively place and remove the icons 22. Such visual transformations more naturally reflect the gradual acquisition of awareness of the comings and goings of workers that workers typically have in physical workspaces.

At any time there may be a large variation in the number of workers who are task proximate to the current worker, as a consequence of the time of day, application base, network size, and other factors. Also, some applications are intended to involve large numbers of different users simultaneously, such as video broadcasting presentation software. In order to accommodate these variations, the user may control the appearance of the encounter window. There are preferably at least two different ways the encounter window 20 can appear when the encounter mechanism is open. For applications where it is anticipated or in fact there are a relatively small number of task proximate workers, the encounter window 20 displays the representations of these users in a compact window format. Figure 1 illustrates the encounter window 20 as it would appear with an open mode of awareness with the compact format.

In the compact format, the encounter window 20 includes the representations of the task proximate workers in a scrollable window pane 24. A scrollable text area 26 provides information about which workers have become task proximate and which have 'left.' In addition, the shared text area 26 allows the workers to communicate with each other by a text dialogue.

The compact format further provides means for obtaining information about the task proximate workers, and a means for contacting one or more workers. In Figure 1, the info button 28, will bring up another window, called a business card 30, for a selected representation of one of the task proximate workers. The business card includes information useful for contacting the worker, such as the worker's telephone extension, mail stop, facsimile number, email address, and so forth. In the preferred embodiment, the business card 30 is a specific view of information about a worker that is stored in a globally accessible database.

Frequently, a worker has multiple applications running, but may not be using the computer for some reason. As one of the goals of the encounter mechanism is to increase the likelihood that the current worker can successfully communicate with task proximate workers,

the encounter mechanism provides information about whether other workers are currently using their computer. One way this information is presented is in the business card 30, which includes text data indicating whether or not that the worker's keyboard, or other input device, is presently active, as shown in Figure 1. If the worker's input device is not active, the encounter mechanism may provide either the length of time since the last activity, or it may provide the time stamp of the last activity.

The business card 30 further provides access to other information about the worker, such as the worker's calendar 32. The current worker may also access information such as the idle time of the worker, and the time since they became task proximate. The business card 30 further is useful to inform the current worker of the identity of other workers with whom a particular worker is interacting, thereby letting the current worker assess the propriety and likelihood of success in attempting to communicate with such worker.

In addition, the current worker can initiate communications with any other worker. Such interactions include posting a text message to the worker owning the business card through the stick-up button 34, or sending email to the worker through the email button 36, or initiating a video conference with the worker through the video-conference button 38. Each of these interaction formats is provided by appropriate services interfaces in the operating system or application framework. In one embodiment, these various services are managed by a communications server that interacts with a object request broker to couple the communications application (such as a video conference server) on one user's computer to a second user's computer.

More particularly, a contact button 40 allows the current worker to initiate an interaction with one or more selected workers. In one embodiment, the current worker selects one or more of the representations in the encounter window, and then presses the contact button 40. The encounter mechanism, in conjunction with a communications server, then selects a

communication mechanism, such as video-conferencing, email, text chat, audio, or the like, depending on the available hardware/software support of each worker. Thus, if both workers have video capability, then a video conference is selected. If only one worker has video capability, and the other, audio support only, then an audio-only dialogue is initiated, and so on.

For applications or contexts in which there are a large number of workers proximate to a current worker, the encounter window 20 may be operated in an extended format. Figure 3 illustrates one embodiment of the extended format. The extended format is useful for applications such as web browsers, databases, or network broadcasting applications where many workers may be using the application simultaneously.

In the extended format, additional functionality is provided to enhance the current worker's ability to find and organize representations of the task proximate workers. In Figure 3, the encounter window 20 is arranged to display a larger number of icons 22, the current worker may switch to a display of names using the radio button 29. This produces the display in Figure 4. The current worker may then sort the set of workers by any number of keys 27, including the worker's name, the time at which the worker became task proximate (either most recent or least recent), the idle times of the workers, or the interaction activity, with workers who are interacting with each other appearing at the top of the window pane. This last sort key enhances the collaborative nature of the encounter mechanism, allowing each worker to see both those other workers who are task proximate and those who are interacting. The worker may also search for specific workers, again, using a variety of search keys, such as personal information, including name, location, department, and the like, or other information, such as activity level, and the like.

The formats of the encounter windows are associated with particular functionality. Since in any application the number of users may vary, the current worker can switch between formats of the encounter window 20 to control functionality, as additional workers "enter" (become task proximate) or "leave" (lose task proximity) the current worker's personal task space. Switching

between formats may be implemented in numerous ways, including re-sizing the encounter window 20 with a suitable user interface element or keystroke.

In addition to controlling the format of the encounter window 20 during the open mode, in one embodiment the worker may also control the sensitivity of the task proximity display, so that the user can distinguish whether a given worker is accessing the same data at the same time, same data at different times, different but related data at the same time, or the like. This allows the user to assess the significance of the task proximity of each worker. These degrees of task proximity may be indicated by different color borders on the icons 22, different border patterns, or positioning of the icons 22, or other visual attributes. For example, using position as a indication of degree of task proximity, the "closer" (more task proximate) workers would placed at the top of the encounter window, and the "farther" (less task proximate) workers would be placed at the bottom.

In some instances, a worker may only want to be peripherally aware of others in the task space in order to concentrate on a particular task or for other reasons. The current worker may then place the encounter mechanism in its minimal mode. In the minimal mode, the encounter mechanism provides an indication only of whether or not at least one other worker is task proximate, but not who the task proximate workers are. This indication may be provided by a simple, relatively small icon. Figures 5a and 5b illustrate an example of the minimal mode, with the minimal mode icon 18 displayed at the top of the window 13 for a web browser application. Figure 5a illustrates the minimal mode icon 18 showing that no other workers are task proximate, with the icon of a person being hollow. The first time the current worker becomes task proximate to another worker, the minimal mode icon 18 is updated to suggest a person being present, as shown in Figure 5b. Other workers entering the current worker's task space do not cause a change in the icon. When the last worker loses task proximity to the current worker the icon 18 reverts to its blank state as shown in Figure 5a. In addition, the minimal mode optionally

includes aural indications commensurate with the visual ones, with distinct tones for when a worker enters the task space and for when a worker leaves the task space.

Finally, the current worker may not be interested in being aware of any other workers at all. Accordingly, a closed mode is provided in which the encounter window 20 is not used at all, and the current worker receives no information about the task proximity of other workers.

In the preferred embodiment, in addition to controlling what a worker sees in their encounter window, the modes further control what other workers see of the particular worker. In the open mode, a worker's representation, whether image, video, or the like, is provided to the encounter windows 20 of other workers who are task proximate. The representation is as shown in Figure 1, with the images of the various workers.

In the minimal mode, a worker desires to be only minimally aware of other workers, and this desire is communicated to other workers who are task proximate. Accordingly, a worker in the minimal mode is seen in other worker's encounter window 20 with an icon or other image representative of the minimal mode, such as a silhouette 42, shown in Figure 1.

Finally, when a worker is in a closed mode, no representation is provided to other task proximate workers, again, consistent with the first worker's intention to not receive or send this state information. Accordingly, the mode of the encounter mechanism is preferably symmetric with respect the information provided to, and received from, other workers. Alternatively, the worker can specify for each mode the type or degree of information to be provided back to other workers.

All the modes may be specified for all applications that a worker is using, or individually for each. For example, a worker may desire to be in an open mode for a web browser, and desire to know which other workers are proximate to the web page, since this may provide further useful information about the activity or task in which the worker is engaged. However, in a tool, such as a spreadsheet or database, or the like, the worker may desire to be in a minimal state,

merely knowing if anyone else is also working on a similar task. Finally, for another application, for example, a word processor, the worker may not want to be disturbed at all.

In another alternate embodiment of the present invention, further information about the availability of workers who are task proximate may be displayed in the encounter window 20 by variations in their icons 22. Availability may be usefully divided into at least five levels: active, idle, engaged, do not disturb, and absent. For each of these levels of availability, there may be assigned a default icon or a user defined icon, each of which visually and distinctly suggests or indicates the associated level of availability. Exemplary icons 22a-e associated with distinct levels of availability are illustrated in Figure 9. The level of availability is determined in one embodiment as a function of the worker's use of their computer, for example, by monitoring their keyboard activity, and using the keyboard inputs per unit of time as a metric for determining the current level of availability. Additionally, the worker's computer and its interconnection with the computer network's communication architecture may be used to determine whether a worker is engaged in an interaction (e.g., telephone, video-conference) with another worker. Procedurally, after a worker is determined to be task proximate, the worker's icon 22 is updated to reflect their current level of availability. As long as the worker remains task proximate, their icon 22 is periodically updated to reflect their current level of availability.

For example, in a typical instance, a second worker will become task proximate to a first worker while being at an active level of availability. Accordingly, the second user's icon 22 will be illustrated in the first user's encounter window 20 with icon 22a, as in Figure 9, or the like. If after a period of time while the second user is still task proximate to the first user, the second user becomes idle, then the icon 22a will be updated to reflect this current level of availability, such as with icon 22b. Later if the second worker then enters into an interaction with another worker (regardless of whether that other worker is task proximate to either the first or second user), then the second worker's icon is again updated, for example to icon 22c.

Preferably, the changes in the level of availability icons 22 are made using gradual visual transformations, such as fades, wipes, dissolves, and the like, to reduce the obtrusiveness of the change. The use and implementation of level of availability information is further described in the related application referenced above.

Architecture of the Encounter Mechanism

Referring now to Figure 6, there is shown a block diagram of one system providing an encounter mechanism. The system includes a number of computers 101 connected on a network 123. Each computer 101 includes a processor 103, an addressable memory 105, a display 107, a local hard disk 109, input/output ports 111, and a network interface 113. Each computer 101 further preferably has coupled to its I/O ports a conventional mouse 119 or other pointing device. Additionally, computers 101 may include audio capability through a microphone 117 and speaker 118, and video capability through a video camera 121. Users connect to the network 123, such as a LAN, WAN, or the like, through the network interface 113, and access remote servers 129, such as naming service, printers 127, storage devices 125, or other computers 101, or remote hosts 131. A suitable computer includes a SPARCstation™ computer manufactured by Sun Microsystems, Inc. of Mountain View, Ca.[1] Any other general purpose computer may also be adapted for use with the invention. Each computer 101 executes a general purpose operating system, such as Sun Microsystems' Solaris™ operating system, with the OpenStep™ windowing environment from Next Computer. Typically, each computer 101 is dedicated to a single worker, though a computer 101 may support multiple workers accessing various different applications as servers to clients on their own computers 101.

[1] Sun and Solaris are trademarks, or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The addressable memory 105 of each computer 101 further includes software components useful for implementing the encounter mechanism for one embodiment of the present invention. Figure 7 illustrates an object model of the architecture of the encounter mechanism.

In the encounter environment, each worker or user has a person object 137. A reference to the person object 137 stores user specific information, such as their user name, account ID, and the like, as conventionally used to create a user handle. The person object 137 further stores the desired representation of the user, as displayed in the encounter window, such as a bit-mapped image, Postscript data, draw object, or text string. These representations may be created automatically or manually, and may be modified by the user. In a preferred embodiment with a large number of users distributed across many different computers 101, there will be a high amount of network communication devoted to periodically updating the encounter windows 143 of each worker. Accordingly, person objects 137 preferably have a compact data structure to reduce the amount of information that is passed between the various components of the encounter mechanism. A reference to the person object 137 is preferably stored in a centralized database 129 such as a naming service which provides a handle to a given user's person object 137 in response to a query presenting the user's name.

In addition, selected data of a person object 137 may be viewed through a business card object 139. In a preferred embodiment, the business card 139 is a view on a person object 137 generated by either the encounter window 143 or other client applications from the person object 137. The business card 139 displays for each user information useful for contacting the user, such as the user's full name, address, email address, telephone number, facsimile number, and the like. In the preferred embodiment, the business cards 139 are extensible, and allow each worker to define one or more new fields for storing additional information about other workers in a locally cached business card 139 component. Such information could be either static data

about the individual (e.g. their pager number) or functional information that facilitates an interaction (e.g. a reference to a service that dials the pager).

The business card object 139 further includes methods for initiating communications with the represented worker. The business card object 139 integrates with existing desktop communications facilities, such as email services, video conferencing, and the like. Generally, in a distributed computing environment a user's local communication client would pass an object reference of a communication server of another worker to an object request broker, which then returns the handle of the communication server to the client. The local client then initiates the communication directly with the communication server. Other mechanisms are also possible. For email communication, for example, the current worker would request the email address of another worker from the person object 137 of that worker, and pass it to a mail tool, which then creates an email message to the recipient.

Also in the preferred embodiment, each person object 137 or business card 139 is a pasteboard type that allows these objects to be cut and pasted between applications, so that users can pass the information about workers between each other through the communication services.

There is further provided on each computer 101 a single encounter window 143. Each encounter window 143 displays representations, such as icons 22, associated with various person objects 137. Across multiple computers 101 there will be multiple encounter windows 143, and so the representation of a single person object 137 may appear in any number of these windows 143. The encounter window 143 provides the functionality described above regarding the modes of awareness, the communication abilities, and the representation of workers therein.

While enabling awareness of, and communication with, other workers is desirable, a worker should be able to control who has access to him in order to maintain and enhance a sense of privacy. In this embodiment, the present invention provides sufficient mechanisms for salient cues of a worker's availability, so that other workers can rely on social mechanisms of protecting

individual privacy. In addition, the specific mechanisms are provided in the encounter window to control which other workers have access to the current worker. The worker may enable or disable various ones of the communication services, such as video-conferencing, to control the degree to which he may be contacted by other workers. In addition, the user can control the level of awareness, by selecting either the open, minimal, or closed mode, preferably with the mode icon 18.

Each person object 137 is indirectly known to one or more encounter-aware applications 131. Generally, each encounter-aware application 131 is used by a single worker at a time. However, in an alternate embodiment, an encounter-aware application 131 may be accessed by multiple users from other computers 101, the application 131 being a server to remote clients.

At any point in time, a user of the application 131 is accessing a particular function and particular data. This information may be represented by file names, object names, pointers, or other means. Each time a user changes the function or data being used, the encounter aware application 131 sends a status message to its encounter proxy object 135. The status message specifies the data or function the worker is currently using, or combination of these elements. The status message may be a string with the name of the data file(s), and a reference to a function, and can include additional information useful to determining each user's position.

Associated with each encounter-aware application 131 is an encounter proxy object 135. The encounter proxy object 135 provides the encounter-aware application 131 a communication mechanism to the encounter server 141. The encounter proxy object 135 accepts from the application 131 a status message describing the user's current position in the application 131. The encounter proxy object 135 is able to obtain a handle to the person object 137 of the worker. The encounter proxy 135 also knows the identity of the application 131 with which it is associated. The encounter proxy object 135 binds the worker's handle, and application identity

with the status message and passes it to the encounter server 141. In this manner, the application 131 does not have to interface directly with the encounter server 141.

The encounter proxy 135 further provides to the encounter server 141 a handle to a match object 133 included in the encounter-aware application 131. This allows the server 141 to directly communicate with the match object 133 and pass to it user position data for a determination of the task proximity of two users. The handle to the match object 133 is provided to the encounter server 141 preferably when the application 131 is executed as a new process.

For each user on each computer 101 there is provided an encounter server 141. The encounter server 141 informs the encounter window 143 of a given user when a representation of person object 137 has to be updated. The encounter server 143 does this according to status messages received from encounter proxy objects 135 on the same computer 101, and from encounter servers 141 on other computers 101. If a user is logged on to two computers 101, there are two encounter servers 141 for the user. Similarly, there may be multiple servers 141 on a single computer for multiple users.

An encounter server 141 maintains a list of the encounter-aware applications 131 on the computer 101, receiving the information about each application 131 from its encounter proxy 135. The server 141 further maintains information identifying which application 131 is currently active (other applications 131 may be operating in the background) for the user. This information is provided by the windowing environment, and is updated from time to time as the active application 131 changes.

The encounter server 141 receives the status messages from the encounter proxy objects 135 on its computer 101, and from other encounter servers 141, and stores these status messages 147. The status messages from the encounter proxy objects 135 identify the application 131, the user's position in the application 131, and the user's handle and position, and a handle to the match object 133 contained in the application 131, if any. In the preferred embodiment, when

the encounter server 141 receives a status message from an encounter proxy object 135, it adds a timestamp to it. The timestamp is useful for ordering the appearance of icons 22 in the encounter window 143, and determining task proximity. The encounter server 141 then sends the status message to all other encounter servers 141 on the network 123.

When the encounter server 141 receives a status message, it compares the received message with stored status messages 147 and identifies status messages that include the same application type or application name, or other matching criteria, and sends the position data included in such status messages to a match object 133 for determining whether the users specified in the status messages are task proximate according to their positions. The encounter server 141 preferably invokes the match object 133 of the currently active application 131.

In a preferred embodiment, the determination of task proximity is made by a match object 133. Each application 131 may have a match object 133 that applies a task proximity rule particular to the type of application. In addition, the encounter server 141 may also have its own library of match objects 133, each one particularly adapted to determining task proximity for different types of applications 131.

Where an application 131 does not have a match object 133, it may indicate to the encounter server 141 the type of match object 133 to use from a library, such as indicating a file name comparison, or a file name and application type, or a file name and time stamp, or any other combination of data. If the application 131 does not indicate the type of match object 133 to use, the encounter server 141 preferably uses a match object 133 that performs a simple string comparison.

Where the application 131 has a match object 133, the application 131 preferably registers the match object with the encounter server 141 through the encounter proxy object 135, with the encounter server 141 receiving and storing a handle to the match object 133. When the

encounter server 141 wants a determination of task proximity, it uses the handle to invoke the match object 133 of the currently active application 131.

In an alternate embodiment, when encounter-aware application 131 starts, its match object 133 is copied down to the encounter server 141, thereby migrating the match object 133 to the encounter server 141. This embodiment supports a relatively fast implementation.

The match object 133 receives from the server 141 two status messages, one describing the user's current position, and one describing the position of another user, typically on a second computer 101. The match object 133 determines whether the users are task proximate to each other, using a predetermined task proximity function. The task proximity function may be a string comparison between the position data for each user, or a more complex function including arithmatic or Boolean comparisons of the time stamps, data files, and the like.

The match object 133 returns an appropriate status value indicating whether or not the two users are task proximate, or alternatively, indicating one of several possible levels of task proximity, as set forth above. With the result from the match object 133, the encounter server 141 informs its encounter window 143. The encounter window 143 then updates its display, either adding a new representation for the task proximate worker, deleting an existing one, or making no change at all.

On each computer 101 there is provided an activity monitor 145. The activity monitor 145 monitors keyboard, mouse, and other input devices to determine whether the worker is currently using the computer 101. If there is no activity for a predetermined length of time, the activity monitor 145 sends a message to the encounter server 141 that the user is idle. This status message includes a handle to the person object of the user, the machine identification number of the user, and an active/idle flag. The encounter server 141 then sends a status message to all other encounter servers 141 on the network 123 indicating the user is not active. The other encounter servers 141 update their encounter windows 143 accordingly. When another worker

brings up the idle worker's business card 139, it will indicate that the worker is not active. When a user becomes active again, the activity monitor 145 sends a status message with an active flag to its encounter server 141, which again broadcasts the status message to the network 123.

An example of the operation of the encounter mechanism is illustrated in Figure 8. Assume two workers, A and B, who may be on separate remote computers or on the same computer. Each worker has an encounter aware application, here an editor 131a, 131b, which contains an encounter proxy object 135 and a match object 133. Each application 131 is preferably built with an application framework that provides notifications when the application 131 becomes active. Each worker has their own encounter server 141a, 141b, and encounter window 143a, 143b. Each encounter server 141 has a handle to the match object 133 of the application 131 being used by the worker associated with the server 141. All communication between the computers 101 is through the encounter servers 141. Each worker is assumed to have their respective encounter windows 143a, 143b in open mode. In this example, the match object 133 is associated with the encounter-aware application 131, though as noted above, it may be associated with the encounter server 141. Also, the activity monitor 145 is not shown for ease of explanation.

Worker B is currently working 800 in his application 131b, as editor, on data file Y. Worker A starts 801 her application 131a, also an editor, making it the active application on her computer. The editor 131a, via the application framework, informs 803 the encounter proxy object 135a that the editor 131a is active. The encounter proxy object 135a sends 807 a status message to the encounter server 141a, indicating that the editor 131a is active. The status message includes the name of the editor 131a, a null position, and a handle to the match object 133a associated with the editor 131a. The position is originally null since worker A has not yet selected a data file to edit. A null position indicates that worker A cannot be task proximate to any other worker at that time, and hence worker A will not appear in any other worker's

encounter window 143, nor will any other worker appear in worker A's encounter window 143a. The handle to the match object 133a is used to register that object with the server 141a.

The encounter server 141a stores 809 as a status message 812a a handle for the editor 131, including the editor name, the match object 133a, and worker A's position. As a worker can have only one application active at a time, this information describes worker A's current application and position. Worker A's encounter server 141a sends 811 a message to the encounter window 143a indicating the current position of worker A. Since the position is null, there is no need to invoke a match object to determine the task proximity of other workers, and so the encounter window 143a clears all current icons 22 or other workers. Storing a handle of the match object 133a enables the encounter server 141a to subsequently invoke that object for determining task proximity between worker A and other workers.

The encounter server 141a also multicasts 813 a status message to all other encounter servers 141 on the network. This status message includes a handle 812 to worker A's person object 137a, an identification number for the computer executing the encounter server 141a, which is worker A's computer, a timestamp generated by the encounter server 141a, the name of the editor 131a, and the position of worker A, which is null.

Worker B's encounter server 141b receives this status message. This encounter server 141b searches 815 its stored status messages 812b, attempting to match the worker A's handle, and worker A's machine identification number to previously received status messages. These stored messages include a status message indicating that worker B is at position Y. If there is a previously stored status message, then this means that worker A may have been task proximate to worker B at some point in the past. Accordingly, the encounter server 141b further determines 817 if the encounter window 143b is displaying a representation of worker A, such as an icon or other image. If so, the encounter server 141b sends a message to the encounter window 143b to remove the representation of worker A, since because of her null position, she is no longer task

proximate to worker B, regardless of what worker B is doing. The encounter window 143b accordingly updates itself. If there was no previously stored message, then the encounter window 143b need not be updated.

The encounter server 143b stores 819 the status message for worker A, replacing any previous status message about worker A. Since, worker A's current position is null, there is no need to determine whether worker A is task proximate to worker B.

Worker A now loads 821 data file X and begins editing it. Worker A's editor 131a sends 803 a message to its encounter proxy object 135a indicating a change in position. In this example, the editor 131 determines task proximity based on identity of data files with a determined time period; here, identity of function within the editor 131 is not required. In this example then, the message from the editor 131a to the encounter proxy object 135a indicates the new position based on data file X, and this position will be called "position X." In other embodiments, the function being used by the editor 131 or other application would be included in the position.

The encounter proxy object 135a sends 807 a status message with the current position X, and editor 131a name to the encounter server 141a. The encounter server 141a searches 805 its stored messages 812a to determine if there are any messages indicating the same application type as editor 131a, or alternatively, application name, for a more limited task proximity function. If a status message is found with the same application type/name, the encounter server 141a invokes 821 the match object 133a to determine whether the position of the worker associated with the located status message is the same as worker A's position, position X. The encounter server 141a passes in the position of the other worker as taken from the found status message.

In this case, the encounter server 141a finds a status message (previously received) indicating that worker B is also using an editor, editor 131b, but with data file Y, and hence at position Y. Again, the editors may have the same name, or the same type. The match object

133a has access to the current position of worker A, position X, since it is associated with editor 131a. The match object 131a determines that position X and position Y are not the same, and returns 823 a task proximity flag so indicating. Thus, worker B does not appear at this point in encounter window 143a, nor does worker A appear in encounter window 143b.

The encounter server 141a sends 813 a status message indicating worker A's current position. Again, this status message includes a handle to worker A's person object 137a, the identification number of worker's computer, a timestamp, the name of editor 141a, and position X. Encounter server 131b receives this message, and replaces 819 the previous status message for worker A with the current one. Since the previous position for worker A was null, there is no need to update the encounter window 143a.

The encounter server 141b compares the new status message, and the application type/name specified therein, with the current application type/name, here editor 131b. These match, and the encounter server 141b invokes 825 the match object 133b associated with the editor 131b, passing in position X from the received status message. The match object 133b compares the positions, position X and position Y, and determines that there is no match, and so informs the encounter server 141b. Accordingly, there is no need to update worker B's encounter window 143b.

Worker A now loads 827 data file Y. Again, the editor 131a informs 803 its encounter proxy object 135a of the new position, position Y, and the encounter proxy object 135a sends 807 a status message with the position information to the encounter server 141a. When the encounter server 141a searches 805 its stored messages, the encounter server 141a identifies a message indicating that worker B also is using an editor 131. The encounter server 141a invokes 821 the match object 133a, which determines that position Y of worker B matches position Y of worker A, returning 823 a task proximity flag to this effect. The encounter server 141a sends 811 a message with a handle to worker B's person object client and representation (icon or the

like) to its encounter window 143a, which then displays the representation. Worker A can now see that worker B is task proximate to her. The timestamp included in the status message may be used by the encounter window 143a to order the appearance of the representations of worker B and others.

The encounter server 141a also sends 813 the status message with worker A's current position Y to other encounter servers 141. The encounter server 141b receives this message, compares 815 it to previous stored messages, and passes 825 the message to the match object 133b along with the position from the previous status message specifying worker B's position. This match object 133b also matches the positions of workers A and B as being task proximate. The encounter server 141b sends 817 a message to the encounter window 143b with the handle to worker A's person object client 137a and icon, and worker B's encounter window 143b updates itself with the representation of worker A.

If worker A desires to interact with worker B, worker A can initiate a video conference or other communication mechanism using one of the buttons on the encounter window 143 or business card of worker B. The encounter server 141a stores, in the status messages, the object reference to worker B and machine identification number of worker B's computer. When requested, the server 141a will use these values to connect to a communication service on worker B's computer.

If a user places their encounter window 143 in the closed mode, their encounter server 141 continues to receive 813 and store status messages. This allows the encounter server 141 to maintain the current state of the mechanism, and thereby update the encounter window 143 if and when the user places their encounter window 143 in the minimal or open mode. In addition, when placed in closed mode, the encounter server 143 sends out a status message with a null position for the user, so that the user is no longer task proximate to any other users. Accordingly, the encounter windows 143 of other users will be updated.

Where the user places the encounter window 143 in the minimal mode, encounter server 141 adds a flag to the status messages it sends out indicating the minimal mode for the worker. When other encounter servers 141 receive this flag in a status message, each encounter server 141 informs its associated encounter window 143 of this flag for a particular user. Each encounter window 143 then updates its display with a suitable representation of the minimal mode, such as the silhouette 42.

Referring again to Figure 7, in the preferred embodiment, person objects 137 are formed in two parts: a lightweight client component, and a more heavyweight server component. The client component provides an address to the server component, and various operations to optimize performance, such as caching the user's representation to reduce the time required to update the encounter window 143, and to connect to the communication services. Pasting the person object 137 generally consists in pasting the address of the server component.

Person object clients are communicated between users, encounter-aware applications, and the encounter windows 143. When a person object client is asked for a value, for example a name, the client accesses the person object 137 server which then returns the value to the client, which in turn forwards it the requesting object. Thus, when the encounter window 143 is updating itself, it will obtain the data for the representation from the server component of the person object 137 if there is not already a representation available in the local person object 137 client. In a preferred implementation person object 137 clients can cache immutable (or very rarely changing) values, for example, the user name, or iconic representation. Other information that depends on executing processes or behaviors are not cached, such as status flag or current activity level. Person object 137 clients preferably provide only read operations to other objects, and do not allow such objects to update the values. The person object 137 server is controlled by the worker whom it represents, and this person is allowed to update values in the server through a suitable interface.

The above description is illustrative and not restrictive. Many variations of the invention will be apparent to those of skill in the art upon review of this disclosure. The scope of the invention should, therefore, be determined with reference to the appended claims, along with their full scope of equivalents, and not merely with reference to the above description.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is an illustration of an encounter window, on a computer display, in open mode, compact format, according to an embodiment of the present invention.

Figure 2 is an illustration of a task space.

Figure 3 is an illustration of the encounter window in open mode format.

Figure 4 is an illustration of another embodiment of the encounter window providing search and sort mechanisms in the open mode, extended format.

Figures 5a and 5b are illustrations of the encounter window in a minimal mode.

Figure 6 is a block diagram of the hardware elements of a computer system supporting the present invention.

Figure 7 is an object model of the system architecture for one aspect of the present invention.

Figure 8 is an event trace illustrating the typical behavior of the encounter mechanism.

Figure 9 is a set of icons illustrating various levels of availability.

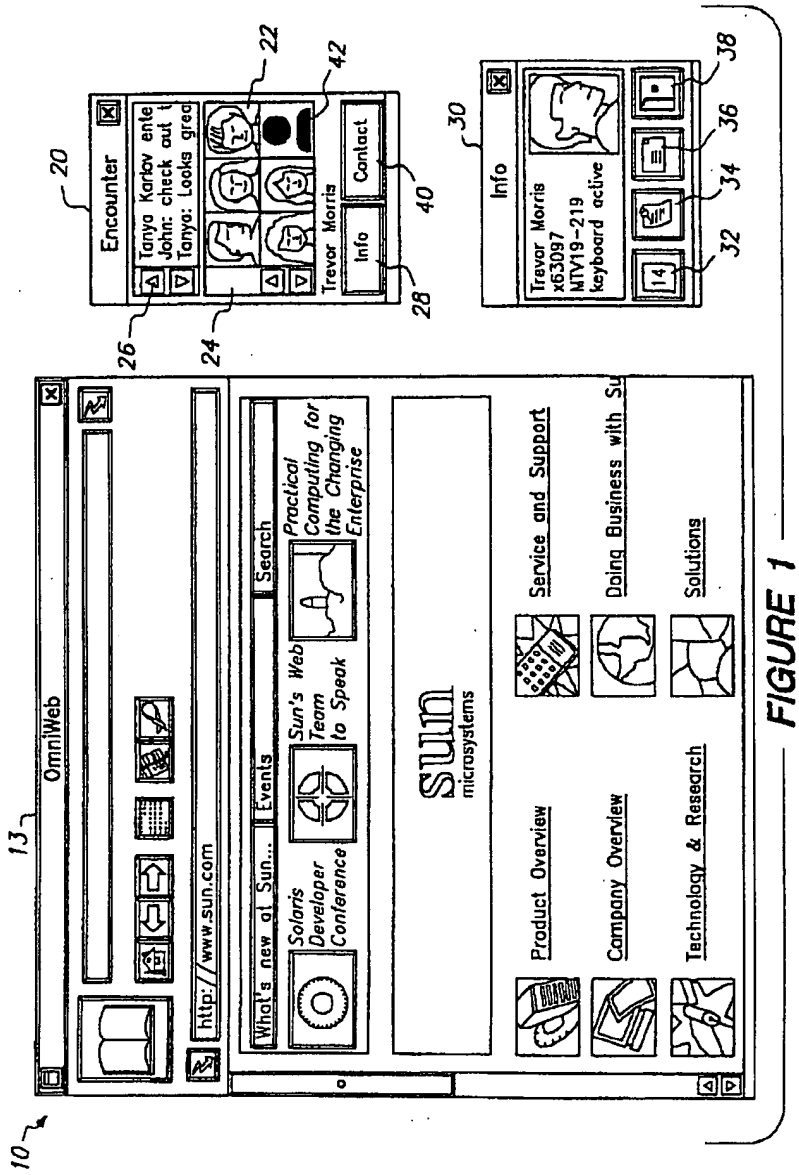

10
13
OmniWeb
http://www.sun.com
What's new at Sun...
Events
Search
Solaris Developer Conference
Sun's Web Team to Speak
Practical Computing for the Changing Enterprise
sun
microsystems
Product Overview
Company Overview
Technology & Research
Service and Support
Doing Business with Su
Solutions
20
Encounter
Tanya Karlov ente
John: check out t
Tanya: Looks grea
22
42
Trevor Morris
Info
Contact
40
28
26
24
30
Info
Trevor Morris
x63097
MTV19-219
keyboard active
38
36
34
32

FIGURE 1

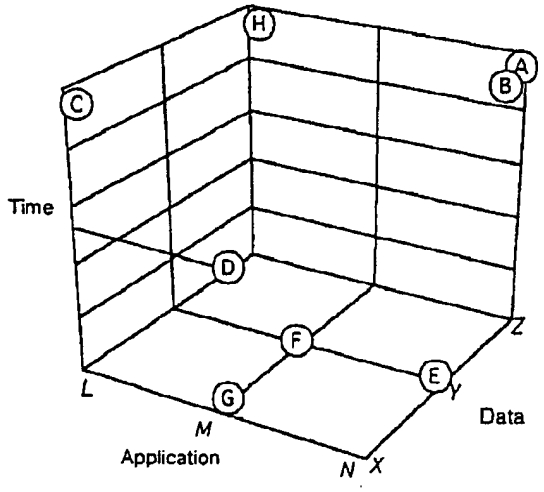


FIGURE 2

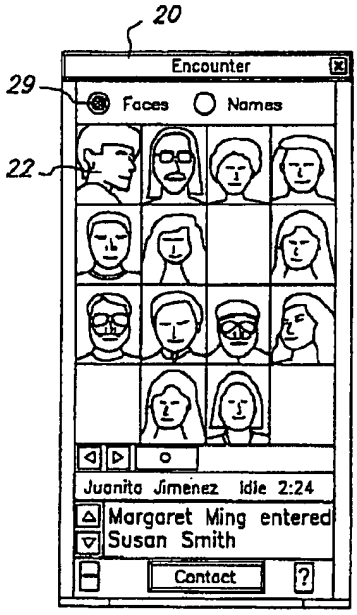


FIGURE 3

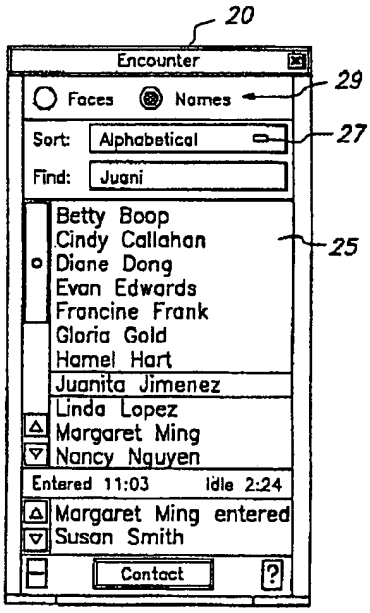


FIGURE 4

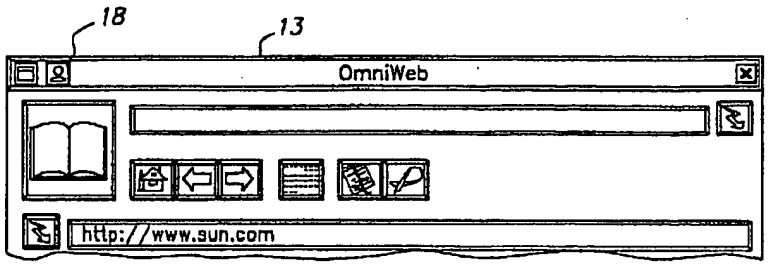


FIGURE 5a

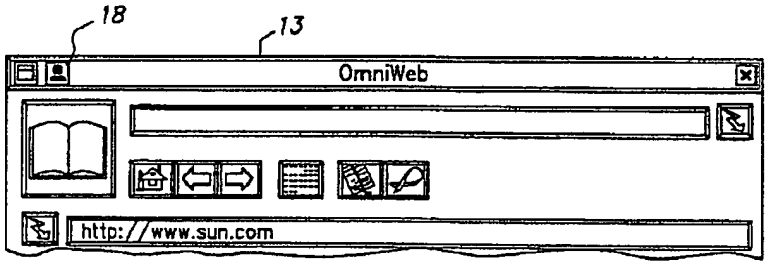


FIGURE 5b

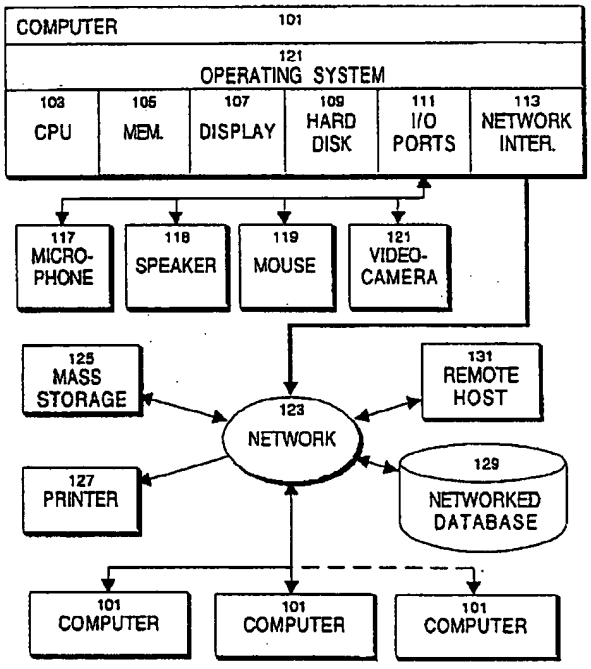


FIGURE 6

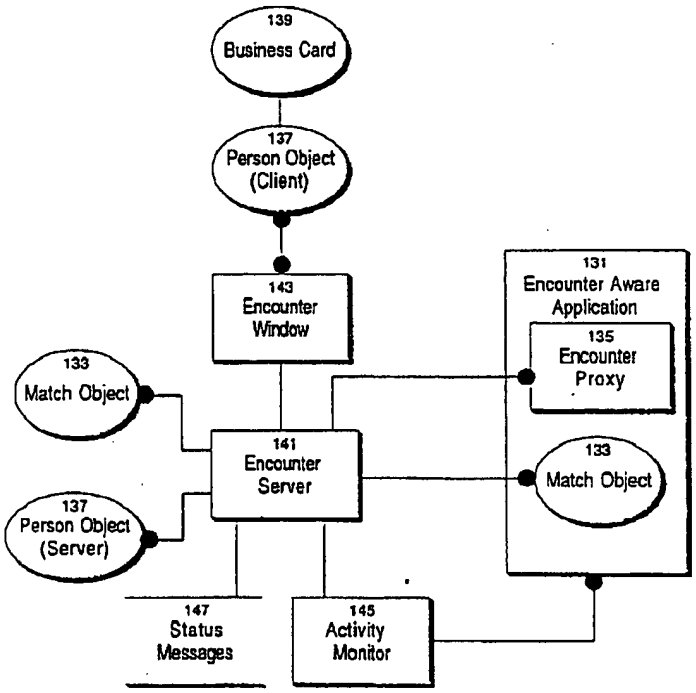

139
Business Card
137
Person Object
(Client)
143
Encounter
Window
131
Encounter Aware
Application
135
Encounter
Proxy
133
Match Object
141
Encounter
Server
133
Match Object
137
Person Object
(Server)
147
Status
Messages
145
Activity
Monitor

FIGURE 7

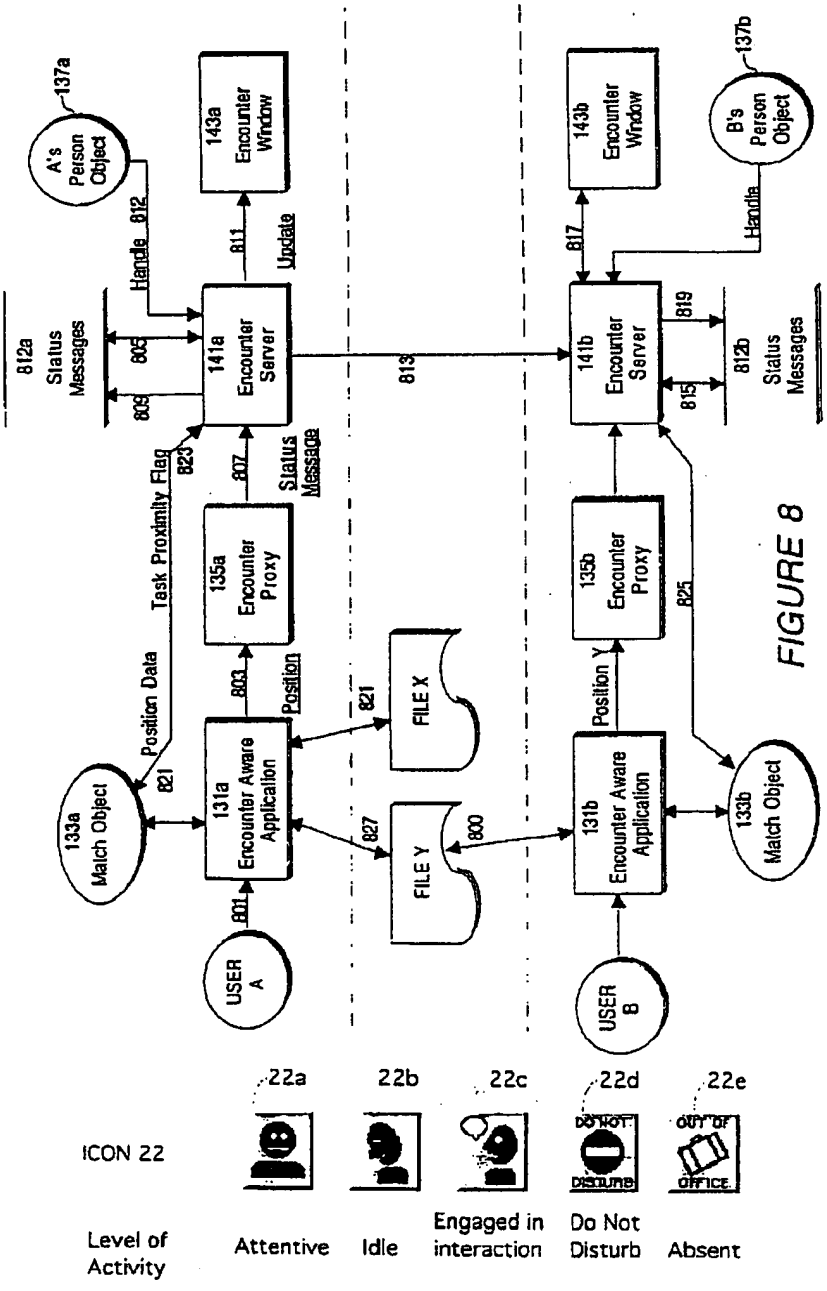


FIGURE 8

FIGURE 9

ABSTRACT OF THE DISCLOSURE

A computer system and method provide networked computer users with information about which other users are task proximate to the user, thereby facilitating spontaneous communications regarding task-related, or other, issues. The information about other users is displayed in a user interface window on each computer that presents a visual representation of each user who is task proximate to the user operating the computer. Task proximity to other users may change as the user context switches between applications, and the user interface window is updated accordingly. Task proximity is determined individually by different applications. One exemplary system architecture for providing the information includes a person object representing each user, and storing the visual representation of the user. An encounter window on each computer displays the visual representations. A number of encounter-aware applications may execute on each computer. An encounter server on each computer provides communication between the encounter-aware applications of the positions of each user, position being determined, for example, by the function the user is using, the data, and the time. At least one encounter-aware application includes a match object that accepts information from the encounter servers about user positions and determines the task proximity of the users. The match object informs the encounter server of the task proximity of the user. The encounter server then updates the appropriate encounter window. The encounter windows further provide a number of communication mechanisms so that users can efficiently contact those other users who are task proximate.

Representative Drawing: Fig. 1

RRS